การเฝ้าสังเกตคุณภาพการให้บริการบนฝั่งเซิร์ฟเวอร์สำหรับเว็บเซอร์วิซ

นายนที อาจเอี่ยม

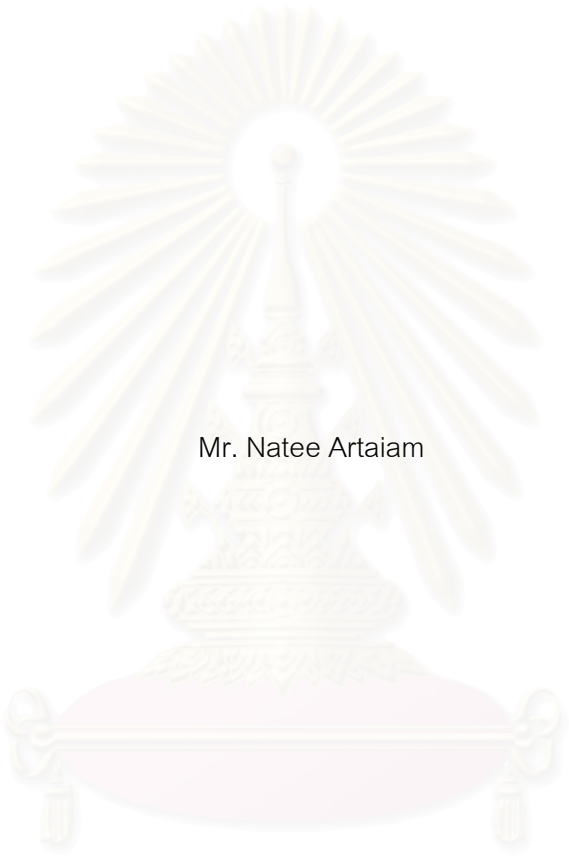
สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2551

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

SERVER-SIDE QOS MONITORING FOR WEB SERVICES

Mr. Natee Artaiam



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2008

Copyright of Chulalongkorn University

หัวข้อวิทยานิพนธ์	การเฝ้าสังเกตคุณภาพการให้บริการบนฝั่งเซิร์ฟเวอร์สำหรับเว็บเซอร์วิซ
โดย	นายนที อาจเอี่ยม
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(อาจารย์ ดร.ยรรยง เต็งอำนวย)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์
(รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

..... กรรมการ
(อาจารย์ จารุมาตร ปิ่นทอง)

..... กรรมการภายนอกมหาวิทยาลัย
(อาจารย์ ดร.เบญจพร ลิ้มธรรมาภรณ์)

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

นที อาจเอี่ยม : การเฝ้าสังเกตคุณภาพการให้บริการบนฝั่งเซิร์ฟเวอร์สำหรับเว็บเซอร์วิซ. (SERVER-SIDE QOS MONITORING FOR WEB SERVICES) อ.ที่ปรึกษาวิทยานิพนธ์ : รศ.ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา, 90 หน้า.

คุณภาพการให้บริการ หรือ คิวโอเอส ถูกนำมาประยุกต์ใช้อย่างกว้างขวางในงานวิจัยและการทำงานที่เกี่ยวข้องกับเว็บเซอร์วิซ เช่น ใช้ในการวิเคราะห์หาคุณภาพและขีดความสามารถในการให้บริการของเว็บเซอร์วิซ หรือ ใช้เป็นข้อมูลเพิ่มเติมในการอธิบายคุณลักษณะของเซอร์วิซที่ได้ประกาศไว้ เป็นต้น อย่างไรก็ตามตัวแบบคิวโอเอสและเครื่องมือที่ใช้ในการเฝ้าสังเกตที่มีในปัจจุบันครอบคลุมหรือรองรับเพียงบางแง่มุมของคุณภาพในการให้บริการของเว็บเซอร์วิซเท่านั้น ในงานวิจัยนี้จะพิจารณาหาตัวแบบคิวโอเอสที่ครอบคลุมหลาย ๆ แง่มุมของคุณภาพในการให้บริการของเว็บเซอร์วิซ โดยเน้นไปที่การเฝ้าสังเกตในฝั่งของผู้ให้บริการเว็บเซอร์วิซเป็นหลัก เนื่องจากสามารถนำข้อมูลที่ได้จากการเฝ้าสังเกตไปใช้ประโยชน์ได้หลายอย่างเช่น นำไปใช้ในวิเคราะห์เพื่อปรับปรุงคุณภาพในการให้บริการของเว็บเซอร์วิซ เป็นต้น โดยแง่มุมที่นำมาใช้ในงานวิจัยนี้ได้แก่ สภาพพร้อมใช้งาน ความสามารถในการเข้าถึงได้ สมรรถนะ ความเชื่อถือได้ ความมั่นคง และการควบคุมดูแลให้เป็นไปตามกฎระเบียบ โดยจะกำหนดตัววัดและข้อมูลจากการเฝ้าสังเกตเพื่อให้ได้มาซึ่งค่าของตัววัดเหล่านั้น นอกจากนี้ ในงานวิจัยยังได้ออกแบบและพัฒนาเครื่องมือเพื่อใช้ในการเฝ้าสังเกตในฝั่งของผู้ให้บริการและคำนวณค่าตัววัดที่ได้กำหนดขึ้น โดยทำการขยายจากจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ซึ่งเป็นซอฟต์แวร์โอเพนซอร์ซ ซึ่งหลังจากทำการพัฒนาและทดสอบแล้ว พบว่าเครื่องมือที่ได้พัฒนาขึ้นสามารถวัดค่าคิวโอเอสได้ถูกต้องตามที่คาดหวังไว้ จะมีเพียงการวัดค่าความมั่นคงเท่านั้นซึ่งเกิดความเบี่ยงเบนจากการวัดค่าผลกระทบจากจุดอ่อนบนเครื่องที่ให้บริการเว็บเซอร์วิซ ตัววัดและเครื่องมือที่พัฒนาขึ้นนี้จะช่วยผู้ให้บริการเว็บเซอร์วิซได้ทราบและเข้าใจถึงการทำงานของเว็บเซอร์วิซของตนในแง่มุมต่าง ๆ และเป็นแนวทางสำหรับนักพัฒนาเว็บเซอร์วิซในการประกาศคุณภาพการให้บริการ หรือพัฒนามอดูลในการเฝ้าสังเกตต่อไป

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต นที อาจเอี่ยม

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออ.ที่ปรึกษาวิทยานิพนธ์หลัก ทวิตีย์ เสนีวงศ์ ณ อยุธยา

ปีการศึกษา 2551

4971433721 : MAJOR COMPUTER SCIENCE

KEYWORDS : QUALITY OF SERVICE / WEB SERVICES / MONITORING

NATEE ARTAIAM : SERVER-SIDE QOS MONITORING FOR WEB SERVICES.

ADVISOR : ASSOC. PROF. TWITTIE SENIVONGSE, Ph.D., 90 pp.

Quality of service (QoS) is adopted widely in Web services research and practice for various purposes such as for metadata publishing and operational management of Web services. Nonetheless, current QoS models and current tools for monitoring support only a limited number of QoS attributes. This research reviews a QoS model which covers various dimensions of service quality and proposes metrics to enhance QoS measurement on the service side. This research focuses on QoS measurement on the service side because the monitored data from the service side can be utilized in various ways such as publishing to service consumers for service selection and service quality improvement etc. The QoS model we utilize in this research are availability, accessibility, performance, reliability, security, and regulatory. A monitoring tool has been designed and implemented as an extension to the Web services monitoring feature of Java System Application Server which is open source software. After implementing and testing the Web service monitoring tool, it is found that the monitoring tool works correctly as expected although there is a small inaccuracy in security measurement which is due to deviation of the measurement of vulnerability impact scores for the Web service-hosting machine. QoS attributes and monitoring tool developed in this research can help service providers to realize their service quality in various dimensions. In addition, they can be used as a guideline for Web service developers to publish their Web service quality or develop monitoring tools.

Department : Computer Engineering Student's Signature : Natee Artaiam

Field of Study : Computer Science Advisor's Signature : Twittie Senivongse

Academic Year : 2008

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี เนื่องมาจากความช่วยเหลืออย่างดียิ่งของท่าน รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้สละเวลาให้คำปรึกษา แนะนำแนวทางเกี่ยวกับงานวิจัยอย่างดีตลอดมาจนเสร็จสมบูรณ์ และผู้วิจัยขอขอบพระคุณคณะกรรมการสอบวิทยานิพนธ์ทุกท่านที่ได้ให้คำแนะนำ ข้อคิดเห็น ข้อเสนอแนะและแนวทางในการพัฒนางานวิจัยนี้

ขอขอบคุณ คุณรัศมีทิพย์ วิตา และพี่ธุรการภาควิชาวิศวกรรมคอมพิวเตอร์ ทุก ๆ คนที่ช่วยอำนวยความสะดวกในการทำงาน และช่วยแนะนำสิ่งดี ๆ เสมอมา

สุดท้ายนี้ ขอขอบพระคุณคุณพ่อ คุณแม่ และพี่สาว สำหรับแรงสนับสนุนและเป็นกำลังใจให้เสมอและขอขอบคุณสิ่งต่าง ๆ ที่ทำให้งานวิจัยสำเร็จลงได้ด้วยดี

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

สารบัญ

หน้า

บทคัดย่อภาษาไทย…….. ง

บทคัดย่อภาษาอังกฤษ..... จ

กิตติกรรมประกาศ..... ฉ

สารบัญ..... ช

สารบัญตาราง……... ฌ

สารบัญภาพ…….. ฎ

บทที่

1 บทนำ…… 1

1.1 ความเป็นมาและความสำคัญของปัญหา……. 1

1.2 วัตถุประสงค์ของการวิจัย……... 8

1.3 ขอบเขตการวิจัย…… 8

1.4 ขั้นตอนการวิจัย……. 8

1.5 ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย…… 8

1.6 โครงสร้างของวิทยานิพนธ์……. 9

2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง……. 10

2.1 ทฤษฎีที่เกี่ยวข้อง……... 10

2.1.1 คุณภาพการให้บริการของเว็บเซอร์วิซที่เป็นที่ต้องการ…….. 10

2.1.2 กลไกการเฝ้าสังเกตเว็บเซอร์วิซของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ (Web Service Monitoring Mechanism of Java System Application Server)..... 12

2.1.3 องค์ประกอบที่จะนำมาใช้ในการเฝ้าสังเกตความมั่นคง…… 13

2.2 งานวิจัยที่เกี่ยวข้อง…… 15

3 การออกแบบตัววัดและเครื่องมือในการเฝ้าสังเกต……... 17

3.1 ตัววัดคุณภาพในการให้บริการ…….. 17

3.2 การออกแบบเครื่องมือที่จะนำมาใช้ในการเฝ้าสังเกต……... 22

4 การพัฒนาเครื่องมือในการเฝ้าสังเกต……... 25

4.1 การพัฒนาส่วนเก็บรวบรวมข้อมูลการเฝ้าสังเกต…….. 25

บทที่ หน้า

4.2 การพัฒนาส่วนของมอดูลวิเคราะห์ความมั่นคง..... 27
4.3 การพัฒนาส่วนของมอดูลในการคำนวณและจัดเก็บค่าตัววัด..... 34
4.4 การพัฒนาส่วนแสดงผลของตัววัดในแอดมินคอนโซล..... 39
5 การทดลองและผลการทดลอง..... 53
5.1 วิธีในการทดลอง..... 53
5.2 ผลการทดลอง..... 57
5.3 วิเคราะห์ผลการทดลอง..... 61
6 สรุปผลการวิจัยและข้อเสนอแนะ..... 68
6.1 สรุปผลการวิจัย..... 68
6.2 ปัญหาและข้อจำกัดของงานวิจัย..... 69
6.3 ข้อเสนอแนะ..... 70
รายการอ้างอิง..... 72
ภาคผนวก..... 74
ภาคผนวก ก วิธีการติดตั้งเครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้น..... 75
ภาคผนวก ข วิธีการติดตั้งเว็บเซอร์วิซในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์..... 79
ภาคผนวก ค ผลงานตีพิมพ์..... 83
ประวัติผู้เขียนวิทยานิพนธ์..... 90

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

สารบัญตาราง

ตาราง หน้า

ตารางที่ 3.1 การเปรียบเทียบแง่มุมของคุณภาพการให้บริการที่เครื่องมือเฝ้าสังเกตเว็บเซอร์วิซรองรับ..... 21

ตารางที่ 4.1 ฟังก์ชันการทำงานใน WebServiceEndpointStatsProvider..... 25

ตารางที่ 4.2 ฟังก์ชันการทำงานในคลาส SecurityAnalyzer 27

ตารางที่ 4.3 ฟังก์ชันที่ให้ข้อมูลในการคำนวณค่าตัววัดของ WebServiceEndPointAggregateStats..... 34

ตารางที่ 4.4 ฟังก์ชันทั้งหมดที่สามารถเรียกใช้ผ่าน WebServiceEndpointAggregateMetrics..... 36

ตารางที่ 4.5 ฟังก์ชันการคำนวณค่าของ WebServiceEndpointAggregateMetrics..... 37

ตารางที่ 4.6 ฟังก์ชันที่ถูกเพิ่มลงไปในคลาส WebServiceHandler 50

ตารางที่ 5.1 ผลการทดสอบการคำนวณตัววัดสภาพพร้อมใช้งาน ที่ได้จากมอดูลในการเฝ้าสังเกตเปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.3. 58

ตารางที่ 5.2 ผลการทดสอบการคำนวณตัววัดความสามารถในการเข้าถึงได้ ที่ได้จากมอดูลในการเฝ้าสังเกตเปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.4..... 58

ตารางที่ 5.3 ผลการทดสอบการคำนวณตัววัดความเชื่อถือได้ ที่ได้จากมอดูลในการเฝ้าสังเกตเปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.5..... 59

ตารางที่ 5.4 ผลการทดสอบการคำนวณตัววัดความมั่นคง ที่ได้จากมอดูลในการเฝ้าสังเกตเปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.6..... 59

ตารางที่ 5.5 ผลการทดสอบการคำนวณตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ ที่ได้จากมอดูลในการเฝ้าสังเกตเปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.7..... 60

ตารางที่ 5.6 ค่าของตัววัดแต่ละชนิดของเว็บเซอร์วิซ ComputeFactorial ที่ได้กำหนดไว้ในแอดมินคอนโซล และค่าของตัววัดแต่ละชนิดที่วัดได้โดยใช้มอดูลในการเฝ้าสังเกต..... 61

ตาราง หน้า

ตารางที่ 5.7 ค่าของตัววัดแต่ละชนิดของเว็บเซอร์วิซ CheckPrimeNumber ที่ได้กำหนดไว้ในแอดมินคอนโซล และค่าของตัววัดแต่ละชนิดที่วัดได้โดยใช้มอดูลในการเฝ้าสังเกต..... 61
ตารางที่ 5.8 สรุปผลการทดสอบความถูกต้องในการทำงานของมอดูลในการเฝ้าสังเกตที่ได้พัฒนาขึ้นในแต่ละกรณีทดสอบของเว็บเซอร์วิซ ComputeFactorial..... 61
ตารางที่ 5.9 สรุปผลการทดสอบความถูกต้องในการทำงานของมอดูลในการเฝ้าสังเกตที่ได้พัฒนาขึ้นในแต่ละกรณีทดสอบของเว็บเซอร์วิซ CheckPrimeNumber..... 62
ตารางที่ 5.10 ผลลัพธ์ในการจับคู่รายการซีวีอีในเครื่องเป้าหมายโดยใช้มอดูลในการเฝ้าสังเกตเทียบกับรายการซีวีอีที่ค้นจากเว็บไซต์ของเอ็นวีดี..... 62

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

สารบัญภาพ

ภาพประกอบ หน้า

รูปที่ 1.1 หน้าจอของโซพนอกซ์..... 2

รูปที่ 1.2 หน้าจอของเว็บอินเจกต์..... 3

รูปที่ 1.3 หน้าจอของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์..... 4

รูปที่ 1.4 หน้าจอของเมเนจเอนจินแอพพลิเคชันเมเนเจอร์..... 5

รูปที่ 1.5 หน้าจอของไมโครซอฟท์เบสไลน์..... 6

รูปที่ 1.6 หน้าจอของแซนด์แคท..... 6

รูปที่ 2.1 องค์ประกอบต่าง ๆ ที่ใช้ในการเฝ้าสังเกตเว็บเซอร์วิซ..... 12

รูปที่ 2.2 กลุ่มตัววัดของซีวีเอสเอส..... 14

รูปที่ 3.1 องค์ประกอบของมอดูลในการเฝ้าสังเกต..... 22

รูปที่ 4.1 รายละเอียดของข้อมูลที่เก็บไว้ในวินโดวส์รีจิสทรี..... 29

รูปที่ 4.2 โครงสร้างของข้อมูลซีวีอีแต่ละตัวในส่วนจัดเก็บข้อมูลเอ็นวีดี..... 30

รูปที่ 4.3 การทำงานของโปรแกรมจับคู่รายการซีวีอีกับรายการซอฟต์แวร์ในเครื่องเซิร์ฟเวอร์ 32

รูปที่ 4.4 รายการซีวีอีที่เผยแพร่ผ่านทางเว็บไซต์ของเอ็นวีดี..... 33

รูปที่ 4.5 ชื่อของไฟล์ข้อมูลรายการซีวีอีและตำแหน่งที่จัดเก็บในไฟล์ ws-metrics-config.xml..... 34

รูปที่ 4.6 โครงสร้างในไฟล์เอ็กซ์เอ็มแอลเพื่อใช้ในการจัดเก็บค่าตัววัด..... 38

รูปที่ 4.7 ความถี่ในการจัดเก็บค่าตัววัดและตำแหน่งที่จัดเก็บในไฟล์ ws-metrics-config.... 39

รูปที่ 4.8 หน้า Statistics..... 40

รูปที่ 4.9 หน้า Messages..... 40

รูปที่ 4.10 หน้า Configuration..... 41

รูปที่ 4.11 หน้า Extension Metrics..... 41

รูปที่ 4.12 หน้า Extension Metrics Configuration..... 42

รูปที่ 4.13 รายละเอียดของสูตรในการคำนวณ..... 43

รูปที่ 4.14 หน้า Regulatory Drill Down..... 44

รูปที่ 4.15 หน้า Vulnerability Information..... 45

รูปที่ 4.16 หน้าต่างแสดงรายละเอียดต่าง ๆ ของซีวีอี..... 46

รูปที่ 4.17 ปุ่ม Update with Online NVD Repository..... 46

รูปที่ 4.18 Extension Metrics Configuration ในหน้า Configuration..... 47

ภาพประกอบ หน้า

รูปที่ 4.19 การทำงานของแอดมินคอนโซลในส่วนที่เกี่ยวข้องกับเว็บเซอร์วิซ..... 49
รูปที่ 4.20 แผนภาพคลาส WebServiceHandler..... 50
รูปที่ 5.1 รายละเอียดของรายการซีวีอี CVE-2008-1412..... 64
รูปที่ 5.2 หน้า Vulnerability Information ในแอดมินคอนโซลที่แสดงรายการซอฟต์แวร์ Windows Mobile 5.0 SDK R2 for Pocket PC 65
รูปที่ 5.3 รายละเอียดของรายการซีวีอี CVE-2008-3873..... 66
รูปที่ 5.4 หน้า Vulnerability Information ในแอดมินคอนโซลที่แสดงรายการซอฟต์แวร์ Adobe Flash Player Plugin..... 66
รูปที่ ก-1 รายละเอียดต่าง ๆ เกี่ยวกับการทำงานของตัวระบบหลังจากพิมพ์คำสั่งเพื่อเริ่มการทำงานของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์..... 76
รูปที่ ก-2 หน้าล็อกอินของแอดมินคอนโซล..... 77
รูปที่ ก-3 หน้าจอของแอดมินคอนโซลหลังจากล็อกอินได้สำเร็จ..... 78
รูปที่ ข-1 รหัสต้นฉบับของคลาส ComputeFactorialService..... 79
รูปที่ ข-2 หน้าจอแสดงรายละเอียดของเว็บเซอร์วิซทั้งหมดที่ถูกติดตั้งในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์..... 80
รูปที่ ข-3 หน้าจอแสดงรายละเอียดต่าง ๆ ของเว็บเซอร์วิซ ComputeFactorialService..... 81
รูปที่ ข-4 หน้าจอแสดงรายละเอียดของไฟล์วิสเดิล (WSDL) ของเว็บเซอร์วิซ..... 81
รูปที่ ข-5 หน้าจอที่ใช้ทดสอบการทำงานของเว็บเซอร์วิซ ComputeFactorialService..... 82
รูปที่ ข-6 ผลลัพธ์จากการทดสอบการทำงานของเว็บเซอร์วิซ ComputeFactorialService... 82

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

คุณภาพการให้บริการ (Quality of Service) หรือ คิวโอเอส (QoS) ถูกนำมาประยุกต์ใช้อย่างกว้างขวางในงานวิจัยและการทำงานที่เกี่ยวข้องกับเว็บเซอร์วิซ ผู้ใช้บริการเว็บเซอร์วิซ (Web Service Consumer) นำคิวโอเอสมาใช้เป็นทางเลือกหนึ่งในการตัดสินใจเลือกเซอร์วิซมาใช้งานในกรณีที่เว็บเซอร์วิซมีการให้บริการเชิงหน้าที่ (Functional Properties) ที่เหมือนๆ กัน ส่วนผู้ให้บริการเว็บเซอร์วิซ (Web Service Provider) นำคิวโอเอสมาใช้ในการวิเคราะห์หาคุณภาพและขีดความสามารถในการให้บริการของตน และใช้เป็นข้อมูลเพิ่มเติมในการอธิบายคุณลักษณะของเซอร์วิซที่ได้ประกาศ (Publish) ใน ยูดีดีไอ (UDDI: Universal Description Discovery and Integration)

อย่างไรก็ตาม คิวโอเอส มักถูกกล่าวถึงในแง่คุณภาพการให้บริการในระบบเครือข่าย (Network System) ซึ่งค่าคิวโอเอสบางประเภทอาจจะไม่เหมาะสมที่จะนำมาใช้ในการวัดคุณภาพการให้บริการของเว็บเซอร์วิซ ในงานวิจัย [1-4] มีการเสนอตัวแบบของคิวโอเอส (QoS Model) ที่แตกต่างกันสำหรับเว็บเซอร์วิซ อย่างไรก็ตามคิวโอเอสที่พบเหมือนกันในตัวแบบเหล่านี้ได้แก่ เวลาในการใช้งาน (Time) ค่าใช้จ่าย (Cost) สภาพพร้อมใช้งาน (Availability) ความเชื่อถือได้ (Reliability) ความมีชื่อเสียง (Reputation) และ ความมั่นคง (Security) ของเซอร์วิซ ในงานวิจัยเหล่านี้บางงานวิจัยกล่าวถึงแต่เพียงคำจำกัดความหรือความหมายของคิวโอเอสนั้นๆ แต่บางงานวิจัยกล่าวถึงวิธีวัดค่าคิวโอเอสนั้นๆ ด้วย นอกจากนี้คิวโอเอสบางชนิดถูกนำเสนอในความหมายที่แตกต่างกันภายใต้ตัวแบบของคิวโอเอสที่แตกต่างกัน ยกตัวอย่างเช่น เวลาในการใช้งานในบางตัวแบบอาจหมายถึงเวลาที่ตอบสนองกลับ (Response Time) ที่วัดในฝั่งของผู้ใช้บริการ แต่ในบางตัวแบบอาจหมายถึงเวลาในการทำงานที่วัดในฝั่งของผู้ให้บริการ เป็นต้น

คิวโอเอสของเว็บเซอร์วิซมักถูกวัดโดยการเฝ้าสังเกต (Monitor) และเก็บรวบรวมข้อมูลที่ได้จากการเฝ้าสังเกตและนำมาคำนวณหาระดับคุณภาพการให้บริการโดยมีพื้นฐานจากตัววัดคิวโอเอส (QoS Metric) ที่คำนวณได้ ในปัจจุบันมีการพัฒนาเครื่องมือที่ใช้ในการเฝ้าสังเกตและเก็บข้อมูลคุณภาพการให้บริการของเว็บเซอร์วิซ ซึ่งเครื่องมือเหล่านั้นได้แก่

1. **โซพนอกซ์ (Soapknox)** [5] เป็นเครื่องมือที่มีจุดเด่นตรงที่ สามารถใช้ในการเฝ้าสังเกตเว็บเซอร์วิซได้หลาย ๆ แพลตฟอร์ม เพราะถูกติดตั้งลงในเครื่องที่ต้องการเฝ้าสังเกตใน

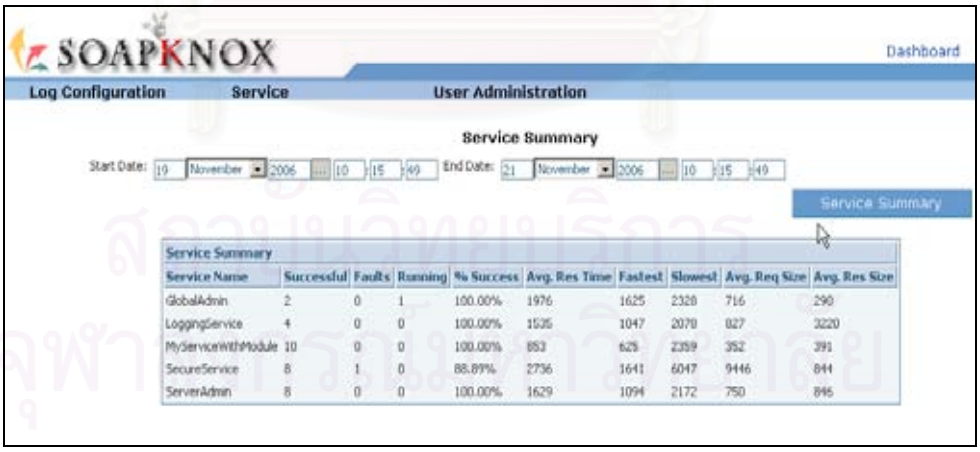
รูปแบบของปลั๊กอิน (Plug-in) โดยทำการเฝ้าสังเกตเว็บเซอร์วิซในฝั่งของผู้ให้บริการ (Service-Side Monitoring) ซึ่งแพลตฟอร์มที่เครื่องมือนี้สามารถใช้ในการเฝ้าสังเกต ได้แก่

- WSO2 Web Services Application Server (WSAS)
- Apache Axis
- Microsoft .NET
- Apache Synapse

ข้อมูลที่โซพนอกซ์เก็บรวบรวมในระหว่างที่ทำการเฝ้าสังเกต ได้แก่ ข้อมูลการเรียกใช้งานเว็บเซอร์วิซ (Service Invocation) รายละเอียดของข้อความโซพ (Simple Object Access Protocol Message) ทั้งในส่วนของหัวเรื่อง (Header) และ เนื้อหา (Body) ของข้อความ โดยจะนำข้อมูลที่ได้มาคำนวณและแสดงผลออกมาในรูปแบบของตัววัดดังนี้

1. ระยะเวลาที่เว็บเซอร์วิซสามารถตอบกลับได้เร็วที่สุด (Fastest Response Time)
2. ระยะเวลาที่เว็บเซอร์วิซสามารถตอบกลับได้ช้าที่สุด (Slowest Response Time)
3. ระยะเวลาที่เว็บเซอร์วิซตอบกลับโดยเฉลี่ย (Average Response Time)
4. ขนาดของข้อมูลที่เว็บเซอร์วิซตอบกลับโดยเฉลี่ย (Average Response Size)
5. อัตราที่เว็บเซอร์วิซจะทำงานได้สำเร็จ (Successful Execution Rate)

ตัวอย่างหน้าจอของโซพนอกซ์แสดงในรูปที่ 1.1



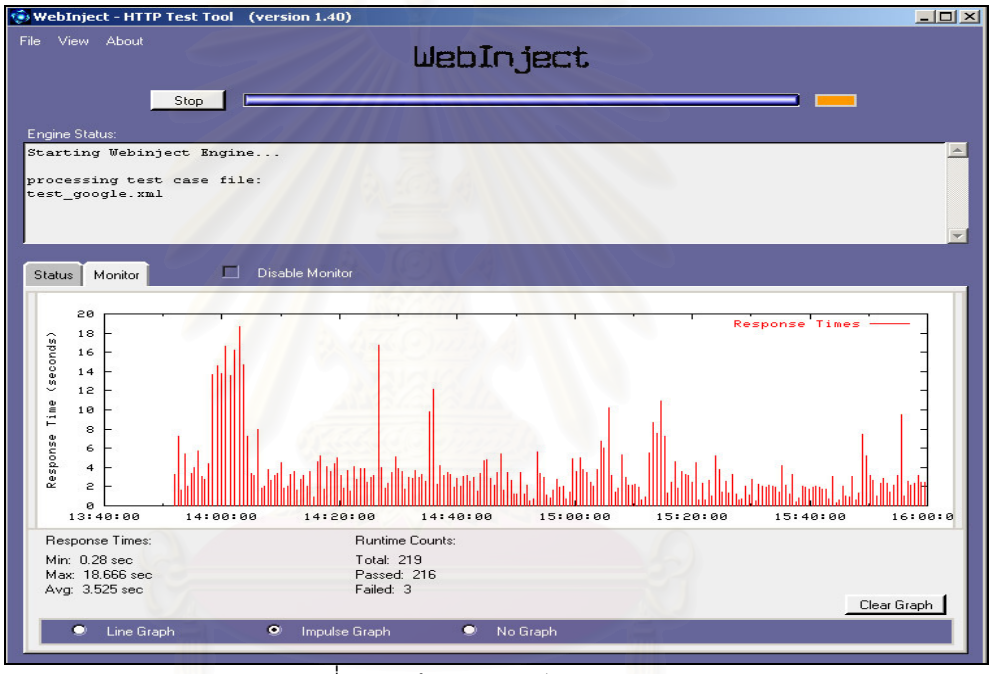
รูป 1.1 หน้าจอของโซพนอกซ์

2. **เว็บอินเจกต์ (WebInject)** [6] เป็นเครื่องมือที่ใช้เฝ้าสังเกตเว็บเซอร์วิซในฝั่งของผู้ใช้บริการ (Client-Side Monitoring) ในการทดสอบและเฝ้าสังเกตโดยการส่งคำร้องขอ (Request) ไปยังเครื่องที่ต้องการเฝ้าสังเกตและทำการวิเคราะห์และรวบรวมข้อมูลจาก

ผลลัพธ์ (Response) ที่ได้รับจากเว็บเซอร์วิซ ซึ่งข้อมูลที่ได้จะถูกคำนวณและแสดงผลออกมาในรูปแบบของตัววัดดังนี้

1. ระยะเวลาที่เว็บเซอร์วิซสามารถตอบกลับได้เร็วที่สุด (Fastest Response Time)
2. ระยะเวลาที่เว็บเซอร์วิซสามารถตอบกลับได้ช้าที่สุด (Slowest Response Time)
3. ระยะเวลาที่เว็บเซอร์วิซตอบกลับโดยเฉลี่ย (Average Response Time)
4. จำนวนคำร้องขอที่ทำงานสำเร็จ (Success Request Counts)
5. จำนวนคำร้องขอที่เกิดความผิดพร่อง (Fault Request Counts)

ตัวอย่างหน้าจอของเว็บอินเจกต์แสดงดังรูปที่ 1.2



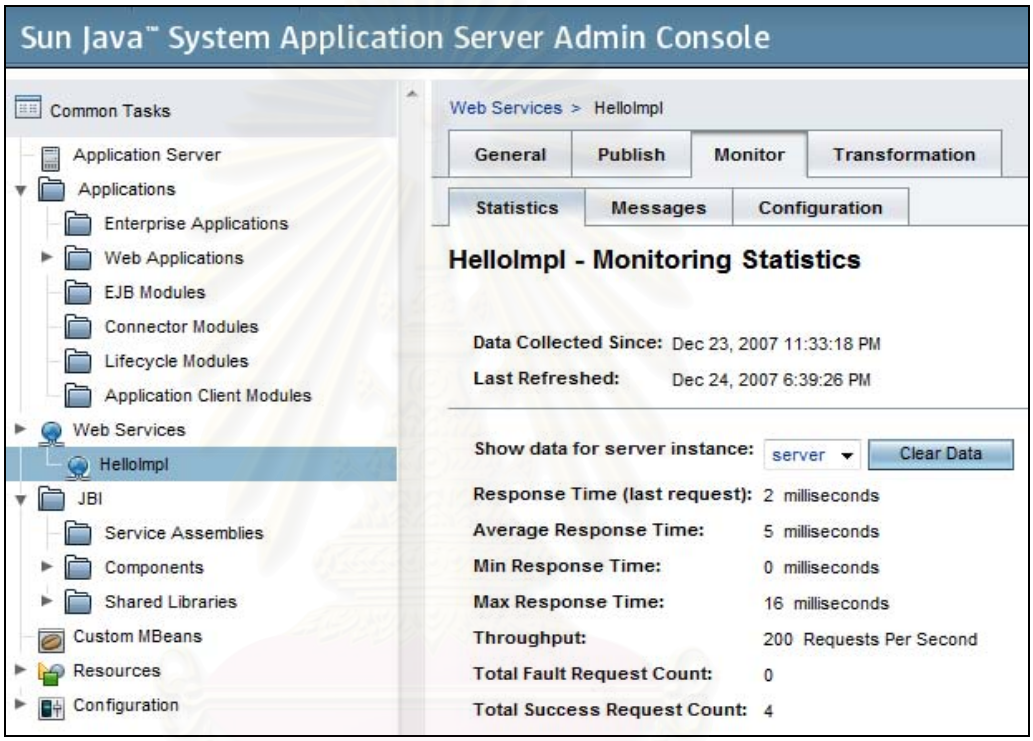
รูปที่ 1.2 หน้าจอของเว็บอินเจกต์

3. **จาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ (Java System Application Server)** [7] เป็นเครื่องมือที่พัฒนาโดยโครงการกลาสฟิช (GlassFish Project) ซึ่งเป็นโอเพนซอร์ซ (Open Source) ของจาวาอีอี 5.0 (Java EE 5.0) โดยจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์เป็นเว็บเซิร์ฟเวอร์ (Web Server) ที่สามารถใช้ในการเฝ้าสังเกตและเก็บข้อมูลของเว็บเซอร์วิซที่ทำงานภายใต้แอพพลิเคชันเซิร์ฟเวอร์ กล่าวอีกนัยหนึ่งคือจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์รองรับการเฝ้าสังเกตในฝั่งของผู้ให้บริการ ซึ่งสามารถเก็บข้อมูลและแสดงผลของข้อมูลที่ได้จากการเฝ้าสังเกตผ่านทางหน้าต่างของแอดมิน (Admin Console) ซึ่งข้อมูลเหล่านั้นได้แก่

1. ระยะเวลาที่เว็บเซอร์วิซสามารถตอบกลับได้เร็วที่สุด (Fastest Response Time)
2. ระยะเวลาที่เว็บเซอร์วิซสามารถตอบกลับได้ช้าที่สุด (Slowest Response Time)

3. ระยะเวลาที่เว็บเซอร์วิซตอบกลับโดยเฉลี่ย (Average Response Time)
4. ปริมาณงานหรือจำนวนการร้องขอต่อวินาที (Throughput)
5. จำนวนคำร้องขอที่ทำงานสำเร็จ (Success Request Counts)
6. จำนวนคำร้องขอที่เกิดความผิดพร่อง (Fault Request Counts)

ตัวอย่างหน้าจอของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์แสดงดังรูปที่ 1.3

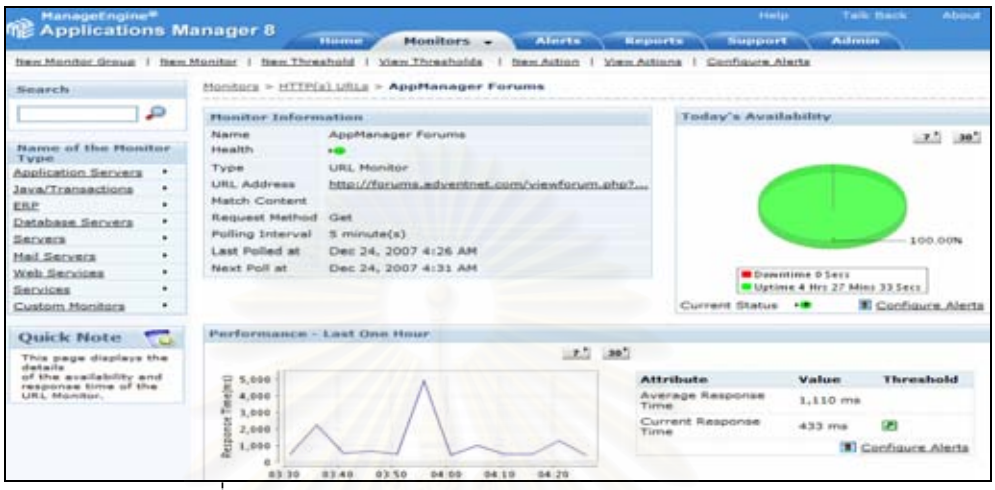


รูปที่ 1.3 หน้าจอของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์

4. **เมเนจเอนจินแอพพลิเคชันเมเนเจอร์** (ManageEngine Application Manager) [8] เป็นเครื่องมือที่ใช้เฝ้าสังเกตเว็บเซอร์วิซในฝั่งของผู้ใช้บริการ สามารถใช้ในการเฝ้าสังเกตและเก็บรวบรวมข้อมูลได้หลาย ๆ ประเภทของเซิร์ฟเวอร์ รวมไปถึงการเฝ้าสังเกตและเก็บรวบรวมข้อมูลของเว็บเซอร์วิซ โดยการส่งคำร้องขอไปยังเครื่องที่ต้องการเฝ้าสังเกตและทำการวิเคราะห์และรวบรวมข้อมูลจากผลลัพธ์ที่ได้รับจากเว็บเซอร์วิซ ซึ่งข้อมูลที่ได้จะถูกคำนวณและแสดงผลออกมาในรูปแบบของตัววัดดังนี้
    1. ระยะเวลาที่เว็บเซอร์วิซตอบกลับมาล่าสุด (Current Response Time)
    2. ระยะเวลาที่เว็บเซอร์วิซตอบกลับโดยเฉลี่ย (Average Response Time)

3. สภาพพร้อมใช้งาน (Availability) ในรูปของอัพไทม์ (Uptime) และ ดาวน์ไทม์ (Downtime)

ตัวอย่างหน้าจอของเมเนจเอนจินแอพพลิเคชันเมเนเจอร์แสดงดังรูปที่ 1.4

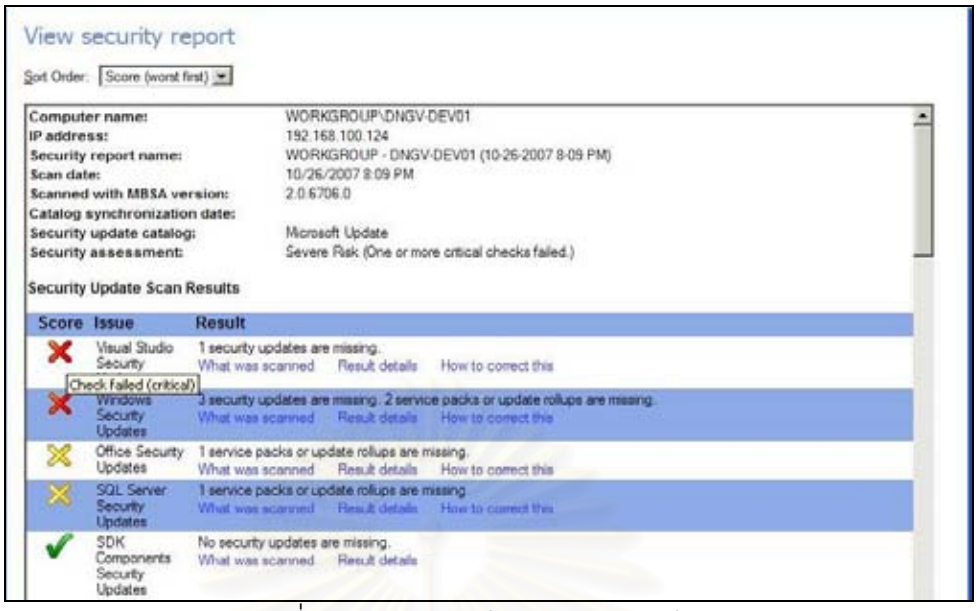


รูปที่ 1.4 หน้าจอของเมเนจเอนจินแอพพลิเคชันเมเนเจอร์

5. **ไมโครซอฟท์เบสไลน์ (Microsoft Baseline Security Analyzer)** [9] เป็นเครื่องมือที่ใช้ในฝั่งของผู้ให้บริการซึ่งสามารถเก็บรวบรวมและวิเคราะห์จุดอ่อน (Vulnerability) ต่าง ๆ ที่พบในเครื่องเว็บเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซนั้น โดยเมื่อลงไมโครซอฟท์เบสไลน์ในเครื่องที่เป็นเว็บเซิร์ฟเวอร์ เครื่องมือนี้จะเก็บรวบรวมข้อมูลต่าง ๆ ของเครื่องซึ่งได้แก่ รายชื่อและเลขเวอร์ชันของโปรแกรมที่ใช้งานในเครื่องเว็บเซิร์ฟเวอร์ รายละเอียดเกี่ยวกับผู้ใช้ที่มีอยู่ในเครื่องและสภาพแวดล้อมอื่น ๆ ภายในเครื่อง หลังจากนั้นจะนำข้อมูลที่ได้ส่งไปวิเคราะห์กับเซิร์ฟเวอร์กลาง (Central Server) และแสดงผลลัพธ์ของการวิเคราะห์พร้อมทั้งคะแนน (Score) ของแต่ละจุดอ่อนที่พบ ซึ่งมีค่าต่าง ๆ ดังต่อไปนี้

1. เป็นตัวอย่างที่ดี (Best Practice)
2. ผ่านการตรวจสอบ (Check Passed)
3. ไม่ผ่านการตรวจสอบแต่ไม่ส่งผลกระทบรุนแรง (Check Failed: Non-critical)
4. ไม่ผ่านการตรวจสอบและส่งผลกระทบรุนแรง (Check Failed: Critical)
5. ข้อมูลในการตรวจสอบไม่เพียงพอ ต้องการข้อมูลเพิ่มเติม (Additional Information)

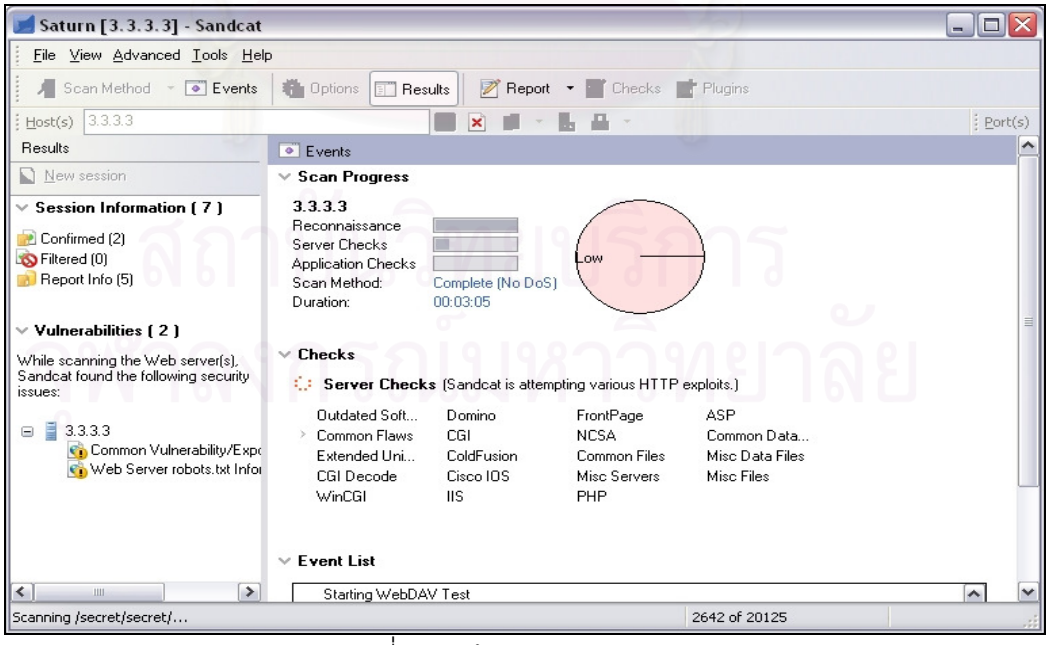
ตัวอย่างหน้าจอของไมโครซอฟท์เบสไลน์แสดงดังรูปที่ 1.5



รูปที่ 1.5 หน้าจอของไมโครซอฟท์เบสไลน์

6. **แซนด์แคท** (Sandcat) [10] เป็นเครื่องมือที่ใช้ในฝั่งของผู้ใช้บริการในการตรวจหา (Scan) จุดอ่อนหรือช่องโหว่ในเว็บไซต์หรือเว็บเซิร์ฟเวอร์ต่าง ๆ โดยแซนด์แคทจะแสดงรายการจุดอ่อนหรือช่องโหว่ที่อยู่ในเว็บไซต์หรือเว็บเซิร์ฟเวอร์นั้นออกมา ซึ่งแบ่งออกเป็น 2 ประเภท คือ จุดอ่อนหรือช่องโหว่ที่ส่งผลกระทบต่อระบบมาก กับ จุดอ่อนหรือช่องโหว่ที่ส่งผลกระทบต่อระบบน้อย

ตัวอย่างหน้าจอของแซนด์แคทแสดงดังรูปที่ 1.6



รูปที่ 1.6 หน้าจอของแซนด์แคท

อย่างไรก็ตามเครื่องมือเหล่านี้ มีการเฝ้าสังเกตและเก็บรวบรวมข้อมูลที่แตกต่างกัน บางเครื่องมือเน้นในส่วนของการเฝ้าสังเกตในเรื่องสมรรถนะของเว็บเซอร์วิซเป็นหลักแต่ยังขาดในส่วนของการเฝ้าสังเกตและเก็บข้อมูลทางด้านความมั่นคง แต่ในบางเครื่องมือเน้นในส่วนของการเฝ้าสังเกตในเรื่องความมั่นคงเป็นหลักแต่ขาดในเรื่องการเฝ้าสังเกตสมรรถนะของเว็บเซอร์วิซ

นอกจากนี้เครื่องมือที่กล่าวข้างต้นยังแบ่งออกเป็น 2 ประเภทหลักๆ ตามลักษณะของการเฝ้าสังเกต ซึ่งได้แก่ การเฝ้าสังเกตในฝั่งของผู้ใช้บริการ และการเฝ้าสังเกตในฝั่งของผู้ให้บริการ การเฝ้าสังเกตในฝั่งของผู้ใช้บริการมักมีเป้าหมายเพื่อนำข้อมูลคุณภาพการให้บริการที่แท้จริงมาประกอบการตัดสินใจเลือกใช้เว็บเซอร์วิซในอนาคตหรือเพื่อประเมินว่าการให้บริการเป็นไปตามข้อตกลงหรือไม่ ส่วนการเฝ้าสังเกตในฝั่งของผู้ให้บริการมีจุดมุ่งหมายที่จะนำข้อมูลเหล่านี้ไปประกาศให้ผู้ใช้บริการได้รับทราบถึงคุณภาพในการให้บริการของตน เช่น นำไปประกาศไว้ในยูดีดีไอ เป็นต้น รวมไปถึงการนำไปใช้วิเคราะห์พฤติกรรมในปัจจุบันของเว็บเซอร์วิซเพื่อนำไปบริหารหรือจัดการต่อไป เช่น ถ้าจากการเฝ้าสังเกตพบว่ามีจำนวนการร้องขอที่ไม่ได้รับการตอบกลับจากเว็บเซอร์วิซเป็นจำนวนมากหรือพบว่าข้อมูลที่ตอบกลับไปมีความผิดพลาด ผู้ให้บริการจะได้ทำการตรวจสอบหาสาเหตุว่าทำไมการร้องขอเหล่านี้จึงไม่ได้รับการตอบกลับ เป็นต้น ทั้งนี้ข้อมูลที่ได้จากการเฝ้าสังเกตทั้ง 2 ประเภทมีความแตกต่างกัน ยกตัวอย่างเช่น สภาพพร้อมใช้งาน เวลาในการประมวลผลที่เว็บเซอร์วิซ (Processing Time) และ จำนวนคำร้องขอทั้งหมดที่มายังเว็บเซอร์วิซเป็นข้อมูลที่สามารถเก็บรวบรวมได้เฉพาะจากการเฝ้าสังเกตในฝั่งของผู้ให้บริการเว็บเซอร์วิซ ส่วนเทิร์นอะราวด์ไทม์ (Turnaround Time) และ เรตติ้งของเว็บเซอร์วิซ (Web Service Rating) เป็นข้อมูลที่สามารถเก็บรวบรวมได้จากฝั่งของผู้ใช้บริการเว็บเซอร์วิซ เป็นต้น

งานวิจัยนี้จะเน้นไปที่การเฝ้าสังเกตในฝั่งของผู้ให้บริการเว็บเซอร์วิซ เนื่องจากสามารถนำข้อมูลที่ได้จากการเฝ้าสังเกตไปใช้ประโยชน์ได้หลายอย่าง และเครื่องมือในฝั่งของผู้ให้บริการซึ่งมีอยู่ในปัจจุบันยังรองรับคิวโอเอสได้ไม่หลากหลายนัก งานวิจัยจะกำหนดตัววัดและข้อมูลจากการเฝ้าสังเกตเพื่อให้ได้มาซึ่งค่าของตัววัดเหล่านั้น โดยยึดตามตัวแบบคิวโอเอสที่ได้ประกาศไว้ใน [4] ซึ่งได้แก่ สภาพพร้อมใช้งาน (Availability) ความสามารถในการเข้าถึงได้ (Accessibility) บูรณภาพ (Integrity) ความเชื่อถือได้ (Reliability) การควบคุมดูแลให้เป็นไปตามกฎระเบียบ (Regulatory) และ ความมั่นคง (Security) นอกจากนี้ในงานวิจัยจะพัฒนาเครื่องมือเพื่อใช้ในการเฝ้าสังเกตและเก็บรวบรวมข้อมูลของเว็บเซอร์วิซในฝั่งของผู้ให้บริการและทำการคำนวณค่าตัววัดที่ได้กำหนดขึ้น โดยทำการขยายจากจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ซึ่งเป็นซอฟต์แวร์โอเพนซอร์ซ

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อนำเสนอตัววัดที่ครอบคลุมแง่มุมต่าง ๆ ในการเฝ้าสังเกตคุณภาพการให้บริการของเว็บเซอร์วิซ
2. เพื่อออกแบบและพัฒนาเครื่องมือในการเฝ้าสังเกต เก็บข้อมูลและคำนวณตัววัดที่กำหนดไว้ในหัวข้อที่ 1

1.3 ขอบเขตของการวิจัย

1. ตัววัดแต่ละตัวที่นำเสนอในงานวิจัยนี้ จะพิจารณาโดยใช้แง่มุมต่าง ๆ ที่อิงตามคุณภาพการให้บริการใน [4]
2. พัฒนามอดูลในการเฝ้าสังเกต เก็บข้อมูลและคำนวณตัววัดบนจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ เวอร์ชัน 2 ยูอาร์ 1 (v2ur1) โดยใช้ภาษาจาวา
3. จุดอ่อนที่จะใช้ในการพิจารณาจะอ้างอิงจากจุดอ่อนที่มาจากรายการของซีวีอีเท่านั้น
4. ฐานข้อมูลที่ใช้ในการค้นหารายการซีวีอีที่เกี่ยวข้องกับซอฟต์แวร์ที่ลงในเครื่องเป้าหมายจะใช้ฐานข้อมูลของเอ็นวีดี
5. คะแนนซีวีเอสเอสที่จะนำมาใช้จะเป็นคะแนนซีวีเอสเอสที่คำนวณโดยเอ็นวีดี ซึ่งคะแนนที่ได้จะเป็นคะแนนที่คำนวณจากกลุ่มตัววัดพื้นฐาน (Base) เท่านั้น

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ส่วนขยายของเครื่องมือสำหรับการเฝ้าสังเกตและเก็บรวบรวมข้อมูลคุณภาพการให้บริการของเว็บเซอร์วิซ บนจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ ที่มีความสามารถมากกว่าเครื่องมือต่าง ๆ ทางฝั่งผู้ให้บริการที่มีอยู่ในปัจจุบัน
2. สามารถนำตัววัดที่ได้ในงานวิจัยนี้ไปใช้เป็นแนวทางสำหรับนักพัฒนาเว็บเซอร์วิซในการประกาศคุณภาพการให้บริการ หรือพัฒนามอดูลในการเฝ้าสังเกตและเก็บรวบรวมข้อมูลคุณภาพการให้บริการของเว็บเซอร์วิซบนแพลตฟอร์มอื่นได้ รวมไปถึงการพัฒนาสำหรับคุณภาพการให้บริการในแง่มุมอื่น ๆ นอกเหนือไปจากที่เสนอไว้ได้

1.5 วิธีดำเนินการวิจัย

1. ศึกษาและรวบรวมแง่มุมต่าง ๆ ในการเฝ้าสังเกตและการเก็บข้อมูลคุณภาพการให้บริการเว็บเซอร์วิซ
2. วิเคราะห์และกำหนดตัววัดเพื่อใช้แทนในแต่ละแง่มุมที่รวบรวมมาจากข้อ 1
3. วิเคราะห์และออกแบบการดึงรายการของซอฟต์แวร์ที่ลงในเครื่องเป้าหมาย รวมไปถึงวิธีในการค้นหาจุดอ่อนที่เกี่ยวข้องกับซอฟต์แวร์เหล่านั้นในฐานข้อมูลเอ็นวีดี
4. ศึกษาโครงสร้างของมอดูลต่าง ๆ และการทำงานของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์

5. วิเคราะห์และออกแบบมอดูลที่ใช้ในการเฝ้าสังเกตและเก็บรวบรวมข้อมูลที่จะสร้างขึ้นในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์
6. วิเคราะห์และออกแบบส่วนของการแสดงผลตัววัดที่ได้จากการคำนวณของมอดูลในข้อ 5
7. พัฒนามอดูลและส่วนของการแสดงผลในข้อ 5 และ 6
8. ทดสอบวิธีการและโปรแกรมที่ได้นำเสนอ
9. วิเคราะห์ผล
10. สรุปผลการวิจัย และจัดทำรายงานวิทยานิพนธ์

1.6 **โครงสร้างของวิทยานิพนธ์**

เนื้อหาของวิทยานิพนธ์ฉบับนี้ถูกแบ่งออกเป็น 6 บท ดังนี้คือ บทที่ 1 เป็นบทนำ บทที่ 2 จะกล่าวถึงทฤษฎีที่จะนำมาประยุกต์ใช้และงานวิจัยที่เกี่ยวข้อง ส่วนบทที่ 3 จะกล่าวถึงการกำหนดตัววัดที่ใช้ในการวัดคุณภาพในการให้บริการและการออกแบบเครื่องมือในการเฝ้าสังเกตเว็บเซอร์วิซ ในบทที่ 4 จะกล่าวถึงการพัฒนาเครื่องมือในการเฝ้าสังเกต ซึ่งจะกล่าวถึงรายละเอียดในการพัฒนาองค์ประกอบต่าง ๆ ของเครื่องมือในการเฝ้าสังเกต ในบทที่ 5 จะเป็นการทดสอบเครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้นและผลที่ได้จากการทดสอบตามชุดการทดสอบต่าง ๆ และในบทสุดท้าย บทที่ 6 จะเป็นการสรุปผลการวิจัยและข้อแนะนำในการพัฒนาต่อไป

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1. คุณภาพการให้บริการของเว็บเซอร์วิซที่เป็นที่ต้องการ

Anbazhagan และ Arun [4] ได้สรุปคุณภาพการให้บริการของเว็บเซอร์วิซที่ผู้ให้บริการเว็บเซอร์วิซควรคำนึงถึง ไว้เป็นหัวข้อใหญ่ ๆ ซึ่งงานวิจัยนี้จะยึดเป็นหลักดังต่อไปนี้

2.1.1.1. **สภาพพร้อมใช้งาน** (Availability) สภาพพร้อมใช้งานเป็นความคาดหวังว่าเว็บเซอร์วิซพร้อมจะให้บริการในช่วงเวลาที่ต้องการ สภาพพร้อมใช้งานของเว็บเซอร์วิซจะอยู่ในรูปของความน่าจะเป็นที่เว็บเซอร์วิซจะพร้อมใช้งานในช่วงเวลาใด เวลาหนึ่ง ถ้ามีค่ามากแสดงให้เห็นว่าเว็บเซอร์วิซมักจะพร้อมให้ใช้บริการอยู่เสมอ นอกจากนี้ยังเกี่ยวข้องกับเวลาในการซ่อมแซม (Time-to-Repair: TTR) ซึ่งแสดงให้เห็นถึงเวลาที่ใช้ในการซ่อมแซมเมื่อเว็บเซอร์วิซไม่สามารถทำงานได้

2.1.1.2. **ความสามารถในการเข้าถึงได้** (Accessibility) ความสามารถในการเข้าถึงได้เป็นคุณสมบัติที่แสดงให้เห็นถึงความสามารถในการให้บริการตามความต้องการของผู้ใช้บริการเว็บเซอร์วิซ ซึ่งสามารถแสดงออกในรูปของค่าความน่าจะเป็นที่เว็บเซอร์วิซจะทำงานได้สำเร็จในช่วงระยะเวลาหนึ่ง ซึ่งอาจจะเกิดสถานการณ์ที่เว็บเซอร์วิซมีสภาพที่พร้อมจะให้บริการแต่ไม่สามารถที่จะเข้าถึงได้ การที่จะให้เว็บเซอร์วิซมีความสามารถในการเข้าถึงสูงสามารถทำได้โดยสร้างระบบที่มีความสามารถในการปรับขนาดตามความต้องการได้สูง (Highly Scalable System) ความสามารถในการปรับขนาดได้ (Scalability) เป็นความสามารถในการให้บริการการร้องขอต่าง ๆ ได้อย่างต่อเนื่องและถูกต้องแม้ว่าจำนวนการร้องขอจะมีมากขึ้นก็ตาม

2.1.1.3. **บูรณภาพ** (Integrity) เป็นความสามารถของเว็บเซอร์วิซในการควบคุมความถูกต้องในการติดต่อระหว่างเว็บเซอร์วิซกับผู้ใช้ ลำดับการทำงานของเว็บเซอร์วิซที่เหมาะสมจากเว็บเซอร์วิชทรานแซกชัน (Web Service Transaction) ที่เข้ามาจะทำให้เกิดความถูกต้องในการติดต่อระหว่างเว็บเซอร์วิซกับผู้ใช้ ซึ่งทรานแซกชัน (Transaction) ก็คือกลุ่มของคำสั่งที่มีลำดับชัดเจนและมองว่าเป็นคำสั่งเพียงคำสั่งเดียว ซึ่งทุกคำสั่งในแต่ละทรานแซกชันจะต้องถูกทำงานจนเสร็จสมบูรณ์จึงจะถือได้ว่าทรานแซกชันนี้ทำงานสำเร็จ ถ้ามีบางคำสั่งในทรานแซกชันไม่ได้ถูกทำงานจนเสร็จ

สมบูรณ์ การเปลี่ยนแปลงทั้งหมดที่เกิดจากคำสั่งในทรานแซกชันนั้นจะถูกยกเลิก (Roll Back)

2.1.1.4. **สมรรถนะ (Performance)** เป็นความสามารถของเว็บเซอร์วิซที่วัดในรูปแบบของปริมาณงาน (Throughput) และ เวลาแฝง (Latency) เว็บเซอร์วิซที่มีประสิทธิภาพดีจะต้องได้ปริมาณงานสูงใช้เวลาแฝงน้อย ปริมาณงานคือจำนวนการร้องขอที่เว็บเซอร์วิซสามารถทำงานได้ในระยะเวลาหนึ่ง เวลาแฝงคือเวลาที่ใช้ตั้งแต่ผู้ใช้บริการส่งการร้องขอไปที่เว็บเซอร์วิซจนกระทั่งได้รับผลลัพธ์จากเว็บเซอร์วิซ ซึ่งอาจเรียกว่า ราวด์ทริปไทม์ (Round-Trip Time)

2.1.1.5. **ความเชื่อถือได้ (Reliability)** เป็นความสามารถของเว็บเซอร์วิซที่แสดงให้เห็นถึงระดับในการดูแลคุณภาพในการบริการ จำนวนของความผิดพลาดหรือความขัดข้อง (Failure) ต่อเดือนหรือต่อปีแสดงให้เห็นถึงความเชื่อถือได้ของเว็บเซอร์วิซนั้น นอกจากนี้ความเชื่อถือได้อาจหมายรวมถึงความสามารถในการส่งและรับข้อความระหว่างผู้ใช้บริการกับเว็บเซอร์วิซซึ่งมั่นใจได้ว่าถูกต้องและเป็นไปตามลำดับ

2.1.1.6. **การควบคุมดูแลให้เป็นไปตามกฎระเบียบ (Regulatory)** เป็นความสามารถของเว็บเซอร์วิซในการควบคุมดูแลให้เป็นไปตามกฎระเบียบหรือเป็นไปตามมาตรฐานที่ได้กำหนดไว้รวมถึงเป็นไปตามเอสแอลเอที่ได้กำหนดขึ้น ซึ่งเว็บเซอร์วิซใช้มาตรฐานหลาย ๆ อย่าง เช่น โซพ (SOAP) ยูดีดีไอ (UDDI) และ วิสเดิล (WSDL) ดังนั้นการใช้เวอร์ชัน (Version) ที่ถูกต้องในแต่ละมาตรฐานที่เว็บเซอร์วิซนำมาใช้ จึงเป็นสิ่งจำเป็นต่อการเรียกใช้บริการอย่างเหมาะสมจากผู้ใช้บริการเว็บเซอร์วิซ

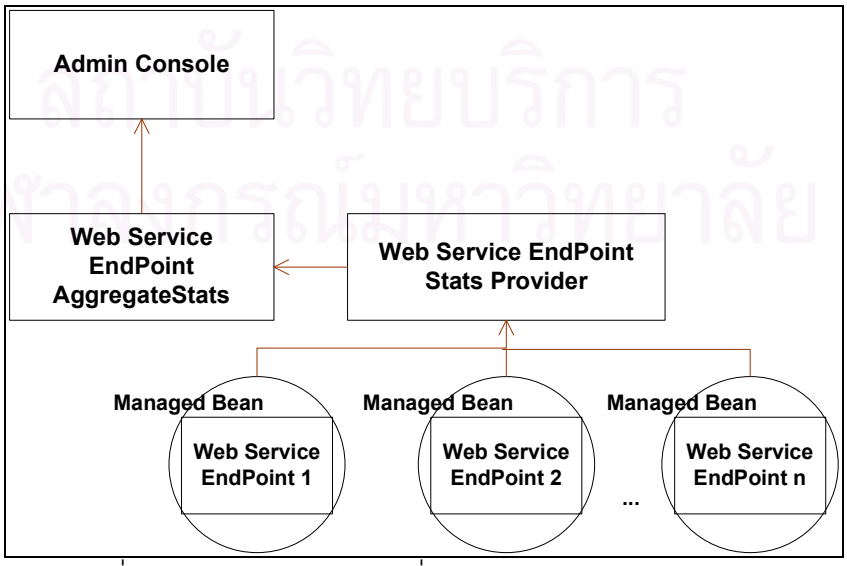
2.1.1.7. **ความมั่นคง (Security)** เป็นความสามารถของเว็บเซอร์วิซในการจัดการเกี่ยวกับความลับของข้อมูล (Confidentiality) และ การไม่ปฏิเสธการร้องขอ (Non-Repudiation) ซึ่งทำได้โดยการพิสูจน์ตัวตนของผู้ที่ใช้บริการ (Authentication) การเข้ารหัสข้อมูล (Encryption) และกระบวนการควบคุมการเข้าถึงข้อมูล (Access Control) ความมั่นคงถูกมองว่าเป็นสิ่งที่สำคัญอย่างหนึ่งเพราะการเรียกใช้บริการเว็บเซอร์วิซกระทำผ่านอินเทอร์เน็ตสาธารณะ (Public Internet) ผู้ให้บริการเว็บเซอร์วิซสามารถใช้หลากหลายวิธีและในหลากหลายระดับในการจัดหาความมั่นคงโดยขึ้นอยู่กับผู้ใช้บริการเป็นหลัก

2.1.2. กลไกการเฝ้าสังเกตเว็บเซอร์วิซของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ (Web Service Monitoring Mechanism of Java System Application Server)

จาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์สามารถเฝ้าสังเกตเว็บเซอร์วิซที่ทำงานภายใต้แอพพลิเคชันเซิร์ฟเวอร์ได้โดยไม่จำเป็นต้องแก้ไขโค้ดในตัวเว็บเซอร์วิซ โดยมีเอเอ็มเอ็กซ์ (AMX: AppServer Management Extension) ทำหน้าที่ในการบริหารจัดการและเฝ้าสังเกตการทำงานของเว็บเซอร์วิซ เอเอ็มเอ็กซ์ถูกพัฒนาขึ้นมาโดยใช้เทคโนโลยีของเจเอ็มเอ็กซ์ (JMX: Java Management Extension)

2.1.2.1 **เจเอ็มเอ็กซ์** [11] เป็นเทคโนโลยีที่นำเสนอความสามารถในการเฝ้าสังเกตหรือการบริหารจัดการโปรแกรมประยุกต์ในภาษาจาวา (Java Application) ในเทคโนโลยีเจเอ็มเอ็กซ์ โปรแกรมประยุกต์ อุปกรณ์ต่าง ๆ (Device) และ บริการต่าง ๆ ในเครื่องคอมพิวเตอร์ (Service on One Machine) ซึ่งจะถูกเรียกรวม ๆ ว่าทรัพยากร (Resource) สามารถถูกควบคุมระยะไกลโดยใช้จาวาบีนอ็อบเจกต์ (Java Bean Object) ที่เรียกว่า เมเนจด์บีน (Managed Bean) หรือ เอ็มบีน (MBean) เอ็มบีนเหล่านี้จะถูกลงทะเบียนไว้ในเซิร์ฟเวอร์ที่ทำหน้าที่บริหารจัดการอ็อบเจกต์เหล่านี้ ซึ่งเรียกว่า เอ็มบีนเซิร์ฟเวอร์ (MBean Server) เอ็มบีนเซิร์ฟเวอร์จะทำหน้าที่เป็นเหมือนตัวแทนในการบริหารจัดการ (Management Agent) ให้กับผู้ใช้งานที่ต้องการเข้าถึงทรัพยากรจากระยะไกล

2.1.2.2 **องค์ประกอบต่าง ๆ ในการเฝ้าสังเกตเว็บเซอร์วิซ** ในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ มีองค์ประกอบต่าง ๆ ที่ใช้ในการเฝ้าสังเกตเว็บเซอร์วิซ ดังรูปที่ 2.1



รูปที่ 2.1 องค์ประกอบต่าง ๆ ที่ใช้ในการเฝ้าสังเกตเว็บเซอร์วิซ

- **เมเนจด์บีน** (Managed Bean) **หรือ เอ็มบีน** (MBean) จากที่ได้กล่าวไปแล้วข้างต้น เอ็มบีนจะห่อหุ้ม (Wrap) แต่ละทรัพยากร ซึ่งในที่นี้คือเว็บเซอร์วิซแต่ละตัวที่ถูกติดตั้งในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ซึ่งถูกเรียกว่า เว็บเซอร์วิซเอนด์พอยนต์ (Web Service EndPoint) เอ็มบีนจะทำหน้าที่เฝ้าสังเกตและเก็บข้อมูลในแต่ละเว็บเซอร์วิซเอนด์พอยนต์ ซึ่งได้แก่ คำร้องขอที่ส่งมาจากผู้ใช้บริการ ตราเวลา (Timestamp) และสถานะของความผิดพร่องที่เกิดขึ้น (Failure Status) เป็นต้น เอ็มบีนจะถูกบริหารจัดการโดย เจเอ็มเอ็กซ์เอเจนต์ (JMX Agent)
- **เว็บเซอร์วิซเอนด์พอยนต์สแตตโพรไวเดอร์** (Web Service EndPoint Stats Provider) เป็นส่วนติดต่อกับผู้ใช้ (Interface) ที่เชื่อมต่อกับอิมพลีเม้นท์คลาสเพื่อเก็บรวบรวมข้อมูลการเฝ้าสังเกตจากแต่ละเมเนจด์บีน เพื่อส่งให้เว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตตนำไปใช้งานต่อ
- **เว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตต** (Web Service EndPoint AggregateStats) เป็นส่วนติดต่อกับผู้ใช้ที่เชื่อมต่อกับอิมพลีเม้นท์คลาสเพื่อดึงข้อมูลจากเว็บเซอร์วิซเอนด์พอยนต์สแตตโปรไวเดอร์ แล้วทำการปรับปรุงข้อมูลให้เหมาะสมเช่น คำนวณค่าตัดวัดบางตัว เช่น ค่าปริมาณงาน (Throughput) ก่อนส่งข้อมูลไปแสดงผลที่แอดมินคอนโซล
- **หน้าต่างของแอดมิน** (Admin Console) เป็นหน้าจอของแอดมิน และเป็นที่ซึ่งข้อมูลที่ได้จากการเฝ้าสังเกตจะถูกนำมาแสดงผลผ่านทางเว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตต นอกจากนี้ยังสามารถปิด/เปิดการเฝ้าสังเกตเว็บเซอร์วิซรวมไปถึงการกำหนดระดับการเก็บข้อมูลที่ได้จากการเฝ้าสังเกตด้วย

2.1.3. องค์ประกอบที่จะนำมาใช้ในการเฝ้าสังเกตความมั่นคง

ในงานวิจัยนี้มีองค์ประกอบสำคัญๆ ที่นำมาใช้ในการเฝ้าสังเกตคุณภาพในการให้บริการในส่วนของความมั่นคงของเว็บเซอร์วิซ ซึ่งองค์ประกอบเหล่านี้ได้แก่

2.1.3.1. **ซีวีอี** (CVE: Common Vulnerabilities and Exposure)

ซีวีอี [12] ก่อตั้งโดย เอ็มไอทีอาร์อี (MITRE) เพื่อจัดหาชื่อที่เป็นมาตรฐานให้กับจุดอ่อน (Vulnerabilities) ในด้านความมั่นคงต่าง ๆ ทั้งนี้เนื่องจากฐานข้อมูลจุดอ่อนมีด้วยกันหลายฐานข้อมูล และแต่ละฐานข้อมูลมีวิธีการตั้งชื่อที่แตกต่างกัน จึงเกิดปัญหาในการอ้างอิงถึงจุดอ่อนเหล่านั้น ซีวีอีจึงเป็นเหมือนดิกชันนารี (Dictionary) ที่กำหนดชื่อที่

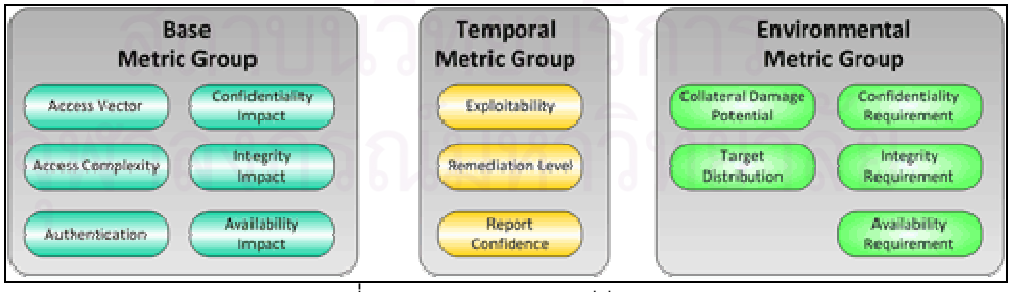
เป็นมาตรฐานให้กับจุดอ่อนต่าง ๆ เพื่อประโยชน์ในการค้นหาหรืออ้างอิงจุดอ่อนในฐานข้อมูลอื่น ๆ ต่อไป

2.1.3.2. **เอ็นวีดี** (NVD: National Vulnerability Database)

เอ็นวีดี [13] เป็นฐานข้อมูลจุดอ่อนจัดตั้งโดยรัฐบาลของประเทศสหรัฐอเมริกา โดยมีจุดมุ่งหมายที่จะจัดหาและรวบรวมข้อมูลจุดอ่อนที่เป็นมาตรฐานเพื่อประโยชน์ในการบริหารจุดอ่อน (Vulnerability Management) และบริหารความมั่นคง (Security Management) โดยอัตโนมัติ เอ็นวีดีประกอบด้วยฐานข้อมูลของรายการทางด้านความมั่นคงที่ควรตรวจสอบ (Security Checklists) ฐานข้อมูลความมั่นคงที่เกี่ยวข้องกับความบกพร่องของซอฟต์แวร์ ฐานข้อมูลเกี่ยวกับชื่อของซอฟต์แวร์ และ ฐานข้อมูลเกี่ยวกับตัววัดผลกระทบ (Impact Metric) ซึ่งในปัจจุบันเอ็นวีดีประกอบด้วยจุดอ่อนภายใต้มาตรฐานการตั้งชื่อของซีวีอีทั้งหมด 28,685 จุดอ่อน

2.1.3.3. **ซีวีเอสเอส** (CVSS: Common Vulnerability Scoring System)

ซีวีเอสเอส [14] เป็นระบบในการให้คะแนนของจุดอ่อนซึ่งถูกออกแบบมาให้จัดหาวิธีที่เป็นมาตรฐานในการให้คะแนนของจุดอ่อนหรือช่องโหว่ที่มีผลกระทบต่อความมั่นคงในระบบเทคโนโลยีสารสนเทศ ซึ่งถ้าจุดอ่อนนี้มีค่าคะแนนซีวีเอสเอสมากแสดงให้เห็นว่าจุดอ่อนนี้ส่งผลกระทบต่อระบบรุนแรงกว่าจุดอ่อนที่มีค่าคะแนนซีวีเอสเอสน้อย ซีวีเอสเอสจะช่วยองค์กรในการจัดลำดับความสำคัญและการตอบสนองต่อจุดอ่อนทางด้านความมั่นคงที่พบได้อย่างเหมาะสม ซีวีเอสเอสประกอบด้วยกลุ่มของตัววัดทั้งหมด 3 กลุ่ม ซึ่งได้แก่ กลุ่มตัววัดพื้นฐาน (Base) กลุ่มตัววัดตามเวลา (Temporal) และกลุ่มตัววัดตามสภาพแวดล้อม (Environmental) ดังรูปที่ 2.2



รูปที่ 2.2 กลุ่มตัววัดของซีวีเอสเอส

- กลุ่มตัววัดพื้นฐาน (Base) เป็นตัวแทนของคุณลักษณะพื้นฐานของจุดอ่อน ซึ่งจะเป็นค่าที่คงที่ไม่เปลี่ยนแปลงในทุกเวลาและทุกสภาพแวดล้อม

- กลุ่มตัววัดตามเวลา (Temporal) เป็นตัวแทนของคุณลักษณะของจุดอ่อนที่มีการเปลี่ยนแปลงอยู่ตลอดเวลาแต่ไม่ขึ้นอยู่กับสภาพแวดล้อม คือมีการเปลี่ยนแปลงตามเวลาอย่างเดียว
- กลุ่มตัววัดตามสภาพแวดล้อม (Environmental) เป็นตัวแทนของคุณลักษณะของจุดอ่อนที่มีการเปลี่ยนแปลงตามสภาพแวดล้อมของผู้ใช้

2.2 งานวิจัยที่เกี่ยวข้อง

ในส่วนของงานวิจัยที่เกี่ยวข้อง จะแบ่งออกเป็น 2 ส่วนคือในส่วนของงานวิจัยที่เกี่ยวข้องกับตัวแบบของคิวโอเอส และ ในส่วนของงานวิจัยที่เกี่ยวข้องกับการเฝ้าสังเกตเว็บเซอร์วิซ

ในส่วนของตัวแบบของคิวโอเอส มีหลาย ๆ งานวิจัยได้นำเสนอตัวแบบของคิวโอเอสซึ่งถูกใช้ในการประกาศเซอร์วิซ หรือการค้นหาเพื่อเลือกเซอร์วิซที่ต้องการมาใช้งาน อย่างไรก็ตามงานวิจัยเหล่านี้ใช้ตัวแบบคิวโอเอสในการจำลองคุณภาพการให้บริการ โดยไม่สนใจว่าค่าคิวโอเอสสามารถคำนวณมาได้อย่างไร ยกตัวอย่างเช่น ตัวแบบคิวโอเอสที่นำเสนอในงานวิจัยของ L. Zeng และคณะ [1] เพื่อใช้ในการเลือกเซอร์วิซมาทำงานร่วมกันในกระแสงาน (Workflow) โดยตัวแบบคิวโอเอสที่นำมาใช้ได้แก่ ค่าใช้จ่ายในการใช้งาน (Execution Price) ช่วงเวลาที่ใช้ในการทำงาน (Execution Duration) ความมีชื่อเสียง (Reputation) อัตราความสำเร็จในการทำงาน (Successful Execution Rate) และ สภาพพร้อมใช้งาน (Availability) ซึ่งค่าของตัวแบบเหล่านี้สามารถเก็บรวบรวมได้จากการเรียกใช้งานเว็บเซอร์วิซ ยกเว้นค่าใช้จ่ายในการใช้งานซึ่งจะถูกกำหนดโดยผู้ให้บริการเว็บเซอร์วิซ และค่าความมีชื่อเสียงซึ่งมาจากการจัดอันดับโดยผู้ใช้บริการเว็บเซอร์วิซ ส่วนในงานวิจัยของ J. Cardoso และคณะ [2] นำเสนอตัวแบบของคิวโอเอสสำหรับใช้ในกระแสงาน โดยประกอบด้วย เวลา (Time) ค่าใช้จ่าย (Cost) ความเชื่อถือได้ (Reliability) และ ฟิเดลิตี (Fidelity) ซึ่งเป็นตัววัดที่แสดงถึงผลการทำงานของเซอร์วิซว่าเป็นไปตามที่ผู้ใช้งานคาดหวังไว้หรือไม่ ในงานวิจัยของ S. Ran [3] นำเสนอยูดีดีไอที่ให้ความสำคัญกับคุณภาพการให้บริการ (Quality-Aware UDDI) ในการค้นหาเซอร์วิซมาใช้งาน ซึ่งได้แก่ สภาพพร้อมใช้งาน ความเชื่อถือได้ ความมั่นคง เป็นต้น ในงานวิจัยของ S. Kalepu และคณะ [15] ได้นำเสนอตัววัดที่ชื่อว่า เวอริตี (Verity) ซึ่งจะนำมาใช้ในการวัดความมีชื่อเสียงของเซอร์วิซ ค่าเวอริตีจะวัดจากระดับความแปรปรวน (Variance) ของคุณภาพการให้บริการที่ประกาศโดยผู้ให้บริการกับคุณภาพในการให้บริการที่วัดได้จริง ในงานวิจัยของ B. Y. Wu และคณะ [16] ได้นำเสนอเวกเตอร์อ้างอิงคิวโอเอส (QoS Reference Vector) ซึ่งใช้ในการอ้างถึงตัวแบบคิวโอเอส ซึ่งได้แก่ ค่าใช้จ่าย เวลา ความเชื่อถือได้ ความไว้วางใจ (Trust) ซึ่งระบุโดยผู้ใช้บริการเว็บเซอร์วิซ และความมั่นคง ซึ่งระบุโดยผู้ใช้บริการเว็บเซอร์วิซเช่นกัน โดยมีค่าอยู่ระหว่าง 1-5

ในส่วนของการเฝ้าสังเกตเว็บเซอร์วิซ งานวิจัยส่วนมากจะเน้นในส่วนของสถาปัตยกรรมของการเฝ้าสังเกตเว็บเซอร์วิซมากกว่าการเพิ่มจำนวนประเภทของคิวโอเอสให้ครอบคลุมคุณภาพการให้บริการที่หลากหลายมากขึ้น ยกตัวอย่างเช่น ในงานวิจัยของ L. Zeng และคณะ [17] ได้ให้ความเห็นว่าตัววัดคิวโอเอส สามารถเฝ้าสังเกตและคำนวณบนพื้นฐานการทำงานของเซอร์วิซ ในงานวิจัยนี้ยังได้นำเสนอระบบการเฝ้าสังเกตคิวโอเอสที่มีสมรรถนะสูง (High-Performance QoS Monitoring System) ซึ่งดำเนินการตามตัวแบบการเฝ้าสังเกตคิวโอเอส โดยมีการกำหนดตัววัดต่าง ๆ ตามระดับในเชิงธุรกิจหรือเชิงเทคโนโลยีสารสนเทศรวมทั้งสูตรที่ใช้ในการคำนวณ ในงานวิจัยของ R. Jurca และคณะ [18] มองว่าคุณภาพในการให้บริการมักมีแนวโน้มที่จะเบี่ยงเบนไปจากค่าที่ได้ประกาศไว้ ดังนั้นจึงได้นำเสนอกลไกในการเฝ้าสังเกตคุณภาพในการให้บริการ (QoS Monitoring Mechanisms) โดยขึ้นอยู่กับผลป้อนกลับ (Feedback) ที่มาจากผู้ใช้บริการเว็บเซอร์วิซในรูปแบบของเรตติ้ง (Rating) เพื่อนำมาคำนวณหาคุณภาพการให้บริการที่แท้จริงต่อไป ส่วนในงานวิจัยของ L. Baresi และคณะ [19] เป็นการสร้างเครื่องมือในการเฝ้าสังเกตเว็บเซอร์วิซที่ทำงานร่วมกันภายใต้บีเพล (BPEL: Business Process Execution Language) ซึ่งเครื่องมือที่สร้างขึ้นสามารถที่จะเฝ้าสังเกตไทม์เอาต์ (Timeout) ความผิดพลาดในระหว่างที่ทำงาน (Run-Time Errors)และ ความเบี่ยงเบนในการทำงาน (Functional Violation) ของเว็บเซอร์วิซที่ทำงานภายใต้บีเพลได้

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 3
การออกแบบตัววัดและเครื่องมือในการเฝ้าสังเกต

ในบทนี้จะกล่าวถึงตัววัดคุณภาพในการให้บริการที่จะนำมาใช้ในการเฝ้าสังเกตรวมถึงข้อมูลที่ต้องการจัดเก็บเพื่อใช้ในการคำนวณตัววัดที่ได้กำหนดขึ้น และกล่าวถึงการออกแบบเครื่องมือที่จะนำมาใช้ในการเฝ้าสังเกตและคำนวณค่าตัววัดที่ได้กำหนดขึ้น ในแง่ของความต้องการโดยรวมของระบบและสถาปัตยกรรมของระบบ

3.1 ตัววัดคุณภาพในการให้บริการ

ในงานวิจัยนี้จะกำหนดตัววัดและข้อมูลที่ต้องการเพื่อให้ได้ข้อมูลของตัววัดนั้น โดยข้อมูลเหล่านี้จะเก็บรวบรวมจากการเฝ้าสังเกตเว็บเซอร์วิซและนำมาคำนวณหาค่าตัววัดแต่ละตัว ซึ่งตัววัดแต่ละตัวก็จะแทนแง่มุมต่าง ๆ ในส่วนของคุณภาพการให้บริการของเว็บเซอร์วิซนั้น โดยยึดตามคุณภาพการให้บริการจากบทความของ Anbazhagan และ Arun [4] ในหัวข้อที่ 2.1.1 ยกเว้นแง่มุมบูรณภาพซึ่งเกี่ยวข้องกับการทำงานของเว็บเซอร์วิซเมื่อร่วมอยู่ในทรานแซกชัน จึงไม่เกี่ยวข้องกับการทำงานของเว็บเซอร์วิซเดี่ยวโดยตรงดังนั้นจึงไม่ได้นำมาใช้ในงานวิจัย ซึ่งตัววัดแต่ละตัวที่ใช้ในงานวิจัยมีดังนี้

3.1.1 **ตัววัดสำหรับสภาพพร้อมใช้งาน** (Availability) – วัดจากค่าความน่าจะเป็นที่เว็บเซอร์วิซจะพร้อมใช้งานในช่วงเวลาใดเวลาหนึ่ง [3] ซึ่งคำนวณได้จาก

$$A_v(s) = T_a(s)/t \qquad (3.1)$$

เมื่อ $A_v(s)$ เป็น ค่าสภาพพร้อมใช้งานของเว็บเซอร์วิซ s

$T_a(s)$ เป็นเวลาทั้งหมด (มีหน่วยเป็นวินาที) ที่เว็บเซอร์วิซ s มีสภาพพร้อมใช้งาน (Up) ภายใต้ระยะเวลาทั้งหมดที่เฝ้าสังเกต

t เป็น ระยะเวลาทั้งหมดที่เฝ้าสังเกตเว็บเซอร์วิซ s (มีหน่วยเป็นวินาที)

3.1.2 **ตัววัดสำหรับความสามารถในการเข้าถึงได้** (Accessibility) – วัดจากจำนวนคำร้องขอทั้งหมดที่เว็บเซอร์วิซสามารถทำงานได้สำเร็จโดยไม่เกิดความผิดพร่อง (Fault) หรือ ไทม์เอาต์ (Time Out) ภายในกรอบเวลาที่คาดหวัง (Expected Time Frame) ในระหว่างช่วงเวลาที่เฝ้าสังเกต เช่น 1 วินาที หรือ 10 วินาที เป็นต้น โดยหากค่าที่ได้จากตัววัดนี้มีค่ามาก ย่อมแสดงว่าเว็บเซอร์วิซสามารถเข้าถึงได้และทำงานได้สำเร็จในกรอบเวลานั้น แม้จะมีจำนวนคำร้องขอเป็นจำนวนมากก็ตาม โดยตัววัดสำหรับความสามารถในการเข้าถึงได้ สามารถคำนวณได้จาก

$$A_c(s) = K(s)/t \quad (3.2)$$

เมื่อ $A_c(s)$ เป็นค่าความสามารถในการเข้าถึงได้ของเว็บเซอร์วิซ s

K(s) เป็นจำนวนคำร้องขอที่เว็บเซอร์วิซ s ทำงานได้สำเร็จโดยไม่เกิดความผิดพร่องหรือไทม์เอาต์ภายในกรอบเวลาที่คาดหวัง ซึ่งกรอบเวลานี้คือเวลาที่ใช้ในการประมวลผลที่ได้ประกาศไว้ (Published Processing Time)

t เป็น ระยะเวลาทั้งหมดที่เฝ้าสังเกตเว็บเซอร์วิซ s (มีหน่วยเป็นวินาที)

ตัววัดสำหรับความสามารถในการเข้าถึงได้จะเหมาะสำหรับการทดสอบการรับภาระงาน (Load Test) ที่มีการส่งคำร้องขอจำนวนมากไปยังเว็บเซอร์วิซและเฝ้าสังเกตจำนวนคำร้องขอที่เว็บเซอร์วิซสามารถทำได้สำเร็จ

3.1.3 **ตัววัดสำหรับสมรรถนะ** (Performance) – เนื่องจากในงานวิจัยจะเน้นการวัดที่ฝั่งของผู้ให้บริการเว็บเซอร์วิซเป็นหลัก ดังนั้นสมรรถนะจึงวัดจาก เวลาที่ใช้กระทำการ (Execution Time) [20] ซึ่งเป็นเวลาทั้งหมดที่เซิร์ฟเวอร์ใช้ในการจัดการคำร้องขอ ซึ่งได้แก่ เวลาที่ใช้ดึงข้อมูลที่ใช้ประมวลผลออกจากคำร้องขอ เวลาที่ใช้ประมวลผลคำร้องขอที่รับเข้ามา และเวลาที่ใช้ในการนำผลที่ได้จากการประมวลผลส่งกลับไปยังผู้ใช้บริการเว็บเซอร์วิซ เป็นต้น ซึ่งตัววัดสำหรับสมรรถนะคำนวณได้จาก

$$T_p(s) = \left[\sum_{i=1}^{N(s)} \Delta T_i \right] / N(s) \quad (3.3)$$

เมื่อ $T_p(s)$ เป็น ค่าเฉลี่ยของเวลาที่ใช้ในการประมวลผลของเว็บเซอร์วิซ s

i เป็น คำร้องขอลำดับที่ i ที่ส่งมายังเว็บเซอร์วิซ s

N(s) เป็นจำนวนคำร้องขอทั้งหมดที่เว็บเซอร์วิซ s ทำงานในช่วงเวลาที่เฝ้าสังเกต

ΔT_i เป็นเวลาที่ใช้ในการประมวลผล (มีหน่วยเป็นมิลลิวินาที) ของคำร้องขอลำดับที่ i ซึ่งคำนวณจากผลต่างระหว่างตราเวลา (Timestamp) ที่ผลลัพธ์ในการประมวลผลถูกส่งกลับไปยังผู้ร้องขอเทียบกับตราเวลาที่คำร้องขอถูกส่งมาที่เว็บเซอร์วิซ s

3.1.4 **ตัววัดสำหรับความเชื่อถือได้** (Reliability) – วัดได้จากอัตราที่เว็บเซอร์วิซจะทำงานได้สำเร็จ (Successful Execution Rate) [1] ซึ่งคำนวณได้จาก

$$R_{rel}(s) = K(s)/N(s) \quad (3.4)$$

เมื่อ $R_{rel}(s)$ เป็น ค่าความเชื่อถือได้ของเว็บเซอร์วิซ s

K(s) เป็นจำนวนคำร้องขอที่เว็บเซอร์วิซ s ทำงานได้สำเร็จโดยไม่เกิดความผิดพร่องหรือไทม์เอาต์ภายในกรอบเวลาที่คาดหวัง โดยกรอบเวลาที่คาดหวังก็คือ เวลาที่ใช้ในการประมวลผลที่ได้ประกาศไว้

N(s) เป็นจำนวนคำร้องขอทั้งหมดที่เว็บเซอร์วิซ s ทำงานในช่วงเวลาที่เฝ้าสังเกต

3.1.5 **ตัววัดสำหรับความมั่นคง** (Security)

ในงานวิจัยนี้จะประเมินความมั่นคงของเว็บเซอร์วิซโดยวัดจากจำนวนจุดอ่อนทั้งหมดที่พบในซอฟต์แวร์ที่ลงไปยังเครื่องที่ต้องการเฝ้าสังเกตและเก็บข้อมูลเช่นเดียวกับงานวิจัย [21] โดยสแกนรายการซอฟต์แวร์ที่มีอยู่ในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ แล้วนำชื่อและเวอร์ชันของซอฟต์แวร์ไปค้นหารายการจุดอ่อนที่เกี่ยวข้องกับซอฟต์แวร์นั้น และจะเพิ่มการพิจารณาผลกระทบของจุดอ่อนทั้งหลายที่มีต่อเครื่องคอมพิวเตอร์โดยใช้ค่าคะแนนซีวีเอสเอส เป็นตัวประเมินผลกระทบที่มีต่อเครื่องคอมพิวเตอร์ที่ให้บริการเว็บเซอร์วิซ และใช้รายการซีวีอีจากฐานข้อมูลเอ็นวีดีเนื่องจาก

1. มีข้อมูลรายการซีวีอี ที่มีการปรับปรุงแก้ไขอยู่เสมอทำให้ข้อมูลมีความทันสมัยอยู่เสมอ และรายการข้อมูลซีวีอีทั้งหมดจะอยู่ในรูปของไฟล์เอ็กซ์เอ็มแอล ทำให้ง่ายในการนำมาใช้เพราะเป็นรูปแบบที่เป็นมาตรฐาน
2. แต่ละรายการซีวีอีจะมีค่าคะแนนซีวีเอสเอสซึ่งบอกค่าความรุนแรงของจุดอ่อนนี้ต่อความมั่นคง กำกับอยู่ในทุกรายการของซีวีอี ซึ่งสามารถนำมาใช้ได้เลยเนื่องจากคำนวณและประเมินจากผู้เชี่ยวชาญและเป็นคะแนนที่ใช้กันแพร่หลาย

การประเมินค่าความมั่นคงของเว็บเซอร์วิซในลักษณะข้างต้นเป็นไปตามแนวคิดที่ว่าผลกระทบที่เกิดจากจุดอ่อนของซอฟต์แวร์ใด ๆ ที่อยู่บนเครื่องคอมพิวเตอร์ที่มีเว็บเซอร์วิซให้บริการอยู่ ถือเป็นผลกระทบที่ส่งผลต่อสภาพแวดล้อมการให้บริการของเว็บเซอร์วิซ ไม่ว่าซอฟต์แวร์นั้นจะเกี่ยวข้องโดยตรงกับเว็บเซอร์วิซหรือไม่ก็ตาม ดังนั้นเมื่อสภาพแวดล้อมของการทำงานไม่มั่นคงย่อมส่งผลต่อความมั่นคงของตัวเว็บเซอร์วิซด้วย

งานวิจัยนี้จะคำนวณตัววัดสำหรับความมั่นคงโดยใช้ค่าเฉลี่ยคะแนนซีวีเอสเอสของรายการซีวีอีทั้งหมดที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ ดังนี้

$$S_e(s) = (\sum_{i=1}^{n} C_i) / n \quad (3.5)$$

เมื่อ $S_e(s)$ เป็น ค่าความมั่นคงของเว็บเซอร์วิซ s

C_i เป็นคะแนนซีวีเอสเอสของรายการซีวีอีลำดับที่ i

n เป็นรายการซีวีอีทั้งหมดที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ

3.1.6 **ตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ** (Regulatory) – วัดโดยคำนวณหาผลต่างระหว่างค่าของตัววัดที่ผู้ให้บริการเว็บเซอร์วิซได้ประกาศไว้กับค่าที่วัดได้จริงจากการเฝ้าสังเกต [22] ซึ่งจะคำนวณได้จาก

$$R_{reg}(s) = \sum_{QoS \in pos} (mQ_i + pQ_i) + \sum_{QoS \in neg} (\frac{1}{mQ_i} - \frac{1}{pQ_i}) \quad (3.6)$$

เมื่อ $R_{reg}(s)$ เป็น ผลต่างระหว่างค่าของตัววัดที่ผู้ให้บริการเว็บเซอร์วิซได้ประกาศไว้กับค่าที่วัดได้ จริงจากการเฝ้าสังเกตเว็บเซอร์วิซ s

mQ_i เป็นค่าตัววัดตัวที่ i ที่ได้จากการเฝ้าสังเกต

pQ_i เป็นค่าตัววัดตัวที่ i ที่ผู้ให้บริการเว็บเซอร์วิซได้ประกาศไว้

pos เป็นกลุ่มของคิวโอเอสเชิงบวก (Positive QoS) ที่ต้องการให้มีค่ามาก ๆ ซึ่งในที่นี้ได้แก่ สภาพพร้อมใช้งาน ความสามารถในการเข้าถึงได้ และความเชื่อถือได้

neg เป็นกลุ่มของคิวโอเอสเชิงลบ (Negative QoS) ที่ต้องการให้มีค่าน้อย ๆ ซึ่งในที่นี้ได้แก่ เวลาที่ใช้ในการประมวลผล และ ค่าผลกระทบจากจุดอ่อนด้านความมั่นคง

ในการคำนวณตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ ค่าของบางตัววัดจำเป็นต้องทำให้เป็นบรรทัดฐาน (Normalize) ก่อนที่จะนำไปคำนวณ ซึ่งได้แก่ ตัววัดความสามารถในการเข้าถึงได้ ซึ่งค่าของตัววัดสามารถมีค่าได้มากกว่า 1 ดังนั้นต้องทำให้เป็นบรรทัดฐานโดยหารค่าที่ได้จากการเฝ้าสังเกตและค่าที่ได้ประกาศไว้ด้วยค่าที่ได้ประกาศไว้ เพื่อให้ค่าอยู่ระหว่าง 0 ถึง 1 ในทำนองเดียวกัน ค่าผลกระทบจากจุดอ่อนด้านความมั่นคงซึ่งมีค่าอยู่ระหว่าง 0-10 ต้องทำการบวกค่าที่ได้จากการเฝ้าสังเกตและค่าที่ได้ประกาศไว้เพิ่มอีกหนึ่ง ให้เป็น 1-11 ก่อนนำไปคำนวณต่อไป

ตารางที่ 3.1 เป็นตารางเปรียบเทียบแง่มุมของคุณภาพการให้บริการที่เครื่องมือเฝ้าสังเกตเว็บเซอร์วิซรองรับ โดยเปรียบเทียบระหว่างเครื่องมือที่มีอยู่ในปัจจุบันกับเครื่องมือที่จะสร้างขึ้นในงานวิจัยนี้

ตารางที่ 3.1 การเปรียบเทียบแง่มุมของคุณภาพการให้บริการที่เครื่องมือเฝ้าสังเกตเว็บเซอร์วิซรองรับ

Tools	Availability	Accessibility	Performance	Reliability	Security	Regulatory
Soapknox	N/A	N/A	Response time	Successful execution rate	Monitor SOAP header	N/A
WebInject	N/A	N/A	Response time	Success/Fault request counts	N/A	N/A
Java System Application Server	N/A	N/A	Response time, Throughput	Success/Fault request counts	Monitor SOAP header	N/A
ManageEngine Application Manager	Uptime, Downtime	N/A	Response Time	N/A	N/A	N/A
Microsoft Baseline Security Analyzer	N/A	N/A	N/A	N/A	List of vulnerabilities and qualitative impacts	N/A
SandCat	N/A	N/A	N/A	N/A	List of vulnerabilities and qualitative impacts	N/A
Our monitoring extension	$A_v(s)$	$A_c(s)$	$T_p(s)$	$R_{rel}(s)$	List of vulnerabilities and $S_e(s)$	$R_{reg}(s)$

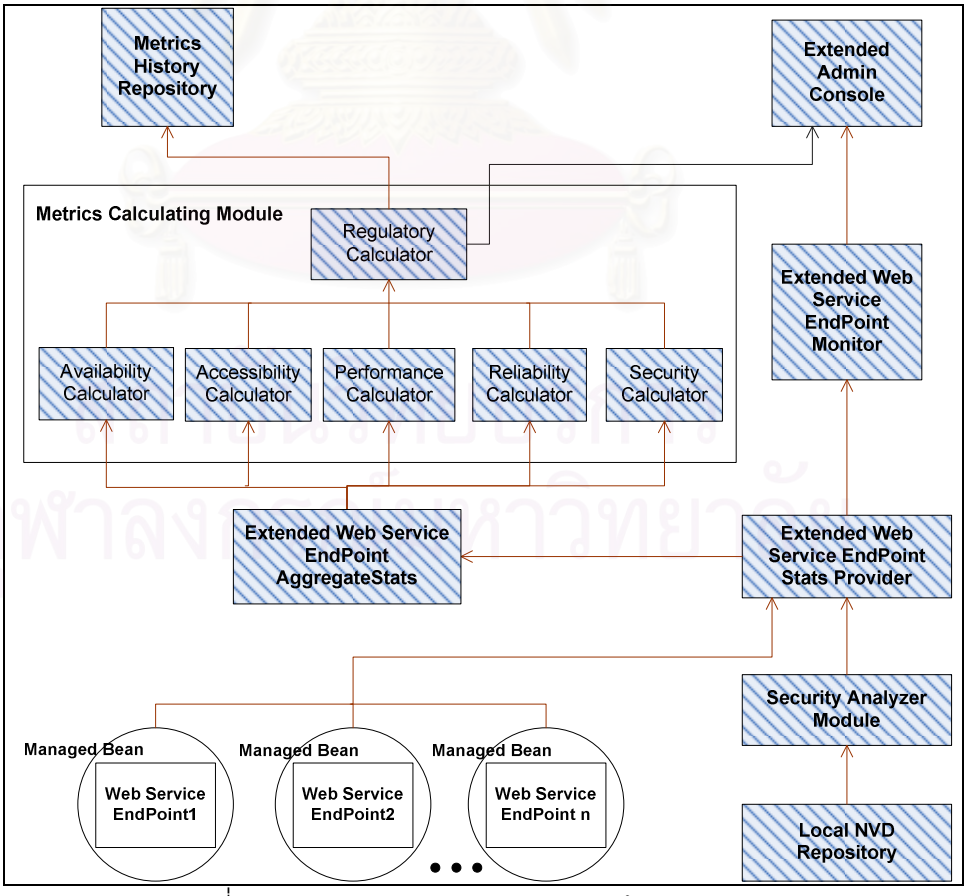
จากตารางจะเห็นได้ว่า เครื่องมือที่จะสร้างขึ้นในงานวิจัยนี้จะครอบคลุมทุกแง่มุมของคุณภาพในการให้บริการที่ระบุไว้ใน [4] (ยกเว้นแง่มุมบูรณภาพซึ่งไม่เกี่ยวข้องกับการทำงานของเว็บเซอร์วิซเดี่ยวโดยตรงและไม่ได้นำมาพิจารณาเช่นเดียวกับเครื่องมืออื่น) โดยในส่วนของตัววัดในแง่มุมสภาพพร้อมใช้งานและความเชื่อถือได้จะเหมือนกับตัววัดของเครื่องมืออื่น ๆ ในตาราง ส่วนตัววัดในแง่สมรรถนะจะแทนด้วยเวลาที่ใช้ในการประมวลผลเช่นเดียวกับโซพนอกซ์และจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ ส่วนตัววัดในแง่ของความมั่นคงจะค้นหารายการของจุดอ่อนที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซเช่นเดียวกับไมโครซอฟท์เบสไลน์และแซนด์แคท แต่จะแตกต่างกับไมโครซอฟท์เบสไลน์และแซนด์แคทตรงที่จะค้นหารายการของจุดอ่อนโดยอิงจากรายการจุดอ่อนในซีวีอีเป็นหลักและจะคำนวณคะแนนโดยใช้วิธีการประเมินของซีวีเอสเอส (ใช้กลุ่มตัววัด

พื้นฐาน) ส่วนตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบจะเป็นตัววัดที่มีเฉพาะในเครื่องมือที่จะสร้างขึ้นในงานวิจัยนี้เท่านั้น

3.2 การออกแบบเครื่องมือที่จะนำมาใช้ในการเฝ้าสังเกต

ในงานวิจัยนี้ ผู้วิจัยจะพัฒนามอดูลที่ในการเฝ้าสังเกตเป็นส่วนขยายของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ ผู้วิจัยเลือกพัฒนามอดูลในการเฝ้าสังเกตบนจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ด้วยเหตุผลที่ว่า ผู้ให้บริการสามารถติดตั้งเว็บเซอร์วิซของตนและได้ประโยชน์จากมอดูลในการเฝ้าสังเกตนี้โดยไม่จำเป็นต้องติดตั้งโปรแกรมอื่น ๆ เพิ่มเติม นอกจากนี้การพัฒนามอดูลในการเฝ้าสังเกตบนจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ยังได้รับประโยชน์จากส่วนที่ทำหน้าที่เฝ้าสังเกตที่มีอยู่แล้วในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ในปัจจุบัน เช่น เวลาที่ใช้ในการตอบกลับ (Response Time) ซึ่งสอดคล้องกับเวลาที่ใช้ในการประมวลผล (Execution Time) ที่เสนอในงานวิจัยนี้ ส่วนจำนวนคำร้องขอทั้งหมดและจำนวนคำร้องขอที่ทำงานสำเร็จสอดคล้องกับข้อมูลที่จะนำมาใช้คำนวณค่าความเชื่อถือได้ เป็นต้น

ในส่วนของการออกแบบมอดูลในการเฝ้าสังเกต องค์ประกอบต่าง ๆ ของมอดูลสามารถแสดงได้ดังรูปที่ 3.1 โดยองค์ประกอบที่ถูกแรเงาในรูปเป็นองค์ประกอบที่จะถูกสร้างขึ้นใหม่ในงานวิจัยนี้ ซึ่งมีรายละเอียดดังนี้



รูปที่3.1 องค์ประกอบของมอดูลในการเฝ้าสังเกต

3.2.1 **ส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์สแตตโพรไวเดอร์** (Extended Web Service EndPoint Stats Provider) เป็นส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์สแตตโพรไวเดอร์ ซึ่งผู้วิจัยทำการขยายเพื่อเพิ่มการเข้าถึงข้อมูลที่ใช้ในการคำนวณตัววัดสภาพพร้อมใช้งาน ตัววัดความสามารถในการเข้าถึงได้ ตัววัดความเชื่อถือได้ และตัววัดสำหรับความมั่นคง

3.2.2 **ส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตต** (Extended Web Service EndPoint AggregateStats) เป็นส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตต ซึ่งต้องมีการขยายส่วนติดต่อผู้ใช้ เพื่อสามารถให้ข้อมูลที่ต้องการไปยังมอดูลคำนวณตัววัดได้ โดยจะดึงข้อมูลจาก ส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์สแตตโพรไวเดอร์ อีกทีหนึ่ง

3.2.3 **ส่วนจัดเก็บข้อมูลเอ็นวีดีภายใน** (Local NVD Repository) เป็นแหล่งที่เก็บรายการซีวีอีทั้งหมดที่มาจากการปรับปรุงข้อมูลล่าสุดกับเว็บไซต์ของเอ็นวีดี ส่วนจัดเก็บข้อมูลเอ็นวีดีภายในจะช่วยอำนวยความสะดวก แม้ในกรณีที่เครื่องที่ถูกเฝ้าสังเกตไม่ได้เชื่อมต่อกับเครือข่ายอินเทอร์เน็ต ก็ยังสามารถใช้ข้อมูลรายการซีวีอีที่มาจากส่วนจัดเก็บข้อมูลเอ็นวีดีภายในแทนได้ อีกทั้งยังช่วยเพิ่มความทนต่อความผิดพร่อง (Fault Tolerance) ให้กับมอดูลวิเคราะห์ความมั่นคง (Security Analyzer Module) ในกรณีที่มอดูลตรวจสอบความมั่นคงไม่สามารถดึงรายการซีวีอีจากเว็บไซต์ของเอ็นวีดีได้ ก็สามารถใช้ข้อมูลจากส่วนจัดเก็บข้อมูลเอ็นวีดีภายในแทนได้

3.2.4 **มอดูลวิเคราะห์ความมั่นคง** (Security Analyzer Module) เป็นมอดูลซึ่งทำหน้าที่ดังต่อไปนี้

- ดึงรายการซอฟต์แวร์ทั้งหมดที่อยู่ในเครื่องที่เว็บเซอร์วิซทำงานอยู่ ซึ่งประกอบด้วยชื่อของซอฟต์แวร์ และเลขเวอร์ชันของซอฟต์แวร์
- ดึงรายการซีวีอีทั้งหมดรวมถึงคะแนนซีวีเอสเอสสำหรับรายการซีวีอีแต่ละตัวจากเว็บไซต์ของเอ็นวีดีซึ่งข้อมูลจะอยู่ในรูปแบบของภาษาเอ็กซ์เอ็มแอล แต่ถ้าไม่สามารถดึงข้อมูลจากเว็บไซต์ของเอ็นวีดีได้ มอดูลจะดึงข้อมูลจากส่วนจัดเก็บข้อมูลเอ็นวีดีภายในแทน
- ปรับปรุงรายการซีวีอีในส่วนจัดเก็บข้อมูลเอ็นวีดีภายในโดยใช้ข้อมูลรายการซีวีอีล่าสุดจากเว็บไซต์ของเอ็นวีดี
- ทำการจับคู่ระหว่างรายการซีวีอีกับรายการซอฟต์แวร์ที่มีชื่อของซอฟต์แวร์และเลขเวอร์ชัน ตรงกับชื่อและเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อนปรากฏตามที่ระบุไว้ในรายการซีวีอีนั้น

- จัดเก็บรายการซีวีอีทั้งหมดที่ได้จากการจับคู่ เพื่อส่งให้ส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์สแตตโพรไวเดอร์ ต่อไป

3.2.5 **มอดูลคำนวณตัววัด** (Metrics Calculating Module) ประกอบด้วยเครื่องคำนวณ (Calculator) ย่อย ๆ ภายในซึ่งจะคำนวณค่าตัววัดต่าง ๆ ตามสูตรที่ได้กล่าวไว้ในหัวข้อ 3.1 โดยจะดึงข้อมูลที่ได้จากการเฝ้าสังเกตจากส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตตเพื่อคำนวณค่าตัววัดคิวโอเอสแต่ละตัวโดยใช้เครื่องคำนวณที่สอดคล้องกับตัววัดเหล่านั้น เช่น สภาพพร้อมใช้งานจะถูกคำนวณจากเครื่องคำนวณตัววัดสภาพพร้อมใช้งาน (Availability Calculator) เป็นต้น นอกจากนี้จากรูปที่ 3.1 จะเห็นได้ว่าเครื่องคำนวณตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบจะได้รับข้อมูลขาเข้าจากเครื่องคำนวณอื่น ๆ ทั้งนี้เนื่องจากค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบคำนวณได้จากค่าตัววัดอื่น ๆ ที่ได้ประกาศไว้กับค่าตัววัดอื่น ๆ ที่ได้จากการเฝ้าสังเกต

3.2.6 **ส่วนจัดเก็บค่าตัววัด** (Metrics History Repository) เป็นไฟล์เอ็กซ์เอ็มแอลที่เก็บข้อมูลของตัววัดแต่ละตัวที่คำนวณไว้ในอดีตเพื่อนำไปใช้ในการวิเคราะห์หรือนำไปบริหารจัดการต่อไป ยกตัวอย่างเช่น การนำข้อมูลที่ได้ไปปรับปรุงค่าคิวโอเอสที่ประกาศไว้ การนำไปวิเคราะห์เพื่อเพิ่มจำนวนเครื่องเพื่อรองรับจำนวนการร้องขออย่างเหมาะสมหรือเพื่อเพิ่มความแข็งแกร่งด้านความมั่นคงให้กับสภาพแวดล้อมของเซอร์วิซ เป็นต้น ซึ่งในหน้าต่างของแอดมินจะสามารถกำหนดให้เก็บข้อมูลตัววัดตามช่วงเวลาที่ต้องการได้

3.2.7 **ส่วนขยายในหน้าต่างของแอดมิน** (Extended Admin Console) เป็นส่วนขยายของแอดมินคอนโซลในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ เพื่อแสดงค่าของตัววัดแต่ละตัวที่ได้จากการคำนวณ นอกจากนี้จะแสดงรายการซอฟต์แวร์ที่ติดตั้งอยู่ในเครื่องที่เป็นเว็บเซิร์ฟเวอร์และรายการจุดอ่อนที่พบ รวมถึงค่าคะแนนซีวีเอสเอสเฉลี่ยของรายการจุดอ่อนซีวีอีที่พบในแต่ละซอฟต์แวร์ด้วย

3.2.8 **ส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์มอนิเตอร์** (Extended Web Service EndPoint Monitor) เป็นส่วนที่ติดต่อกับผู้ใช้ ที่อนุญาตให้แอดมินสามารถแก้ไขโครงแบบ (Configuration) ของเครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้นให้เหมาะกับการใช้งาน โดยเชื่อมต่อกับส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์สแตตโพรไวเดอร์ เพื่อให้แอดมินสามารถเข้าถึงข้อมูลหรือควบคุมการทำงานของ ส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์สแตตโพรไวเดอร์ได้

บทที่ 4
การพัฒนาเครื่องมือในการเฝ้าสังเกต

ในบทนี้จะกล่าวถึงการพัฒนาเครื่องมือในการเฝ้าสังเกตและคำนวณค่าตัววัดประเภทต่าง ๆ ตามที่ได้ทำการออกแบบไว้ในบทที่ 3 โดยได้แบ่งการพัฒนาออกเป็น 4 ส่วนหลัก ๆ ซึ่งได้แก่ ส่วนเก็บรวบรวมข้อมูลการเฝ้าสังเกต ส่วนของมอดูลวิเคราะห์ความมั่นคง ส่วนของมอดูลในการคำนวณและจัดเก็บค่าตัววัด และส่วนแสดงผลของตัววัดในแอดมินคอนโซล

4.1 การพัฒนาส่วนเก็บรวบรวมข้อมูลการเฝ้าสังเกต

เนื่องจากในการคำนวณตัววัดในหัวข้อ 3.1.1 ต้องมีการเก็บข้อมูลเพิ่มเติมนอกเหนือจากข้อมูลที่จาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์จัดหาให้ ดังนั้นต้องเขียนโปรแกรมเพิ่มเติมในคลาส WebServiceEndpointStatsProviderImpl ของเว็บเซอร์วิซเมเนจบีน เพื่อเก็บข้อมูลที่ใช้ในการคำนวณตัววัดสภาพพร้อมใช้งาน ตัววัดความสามารถในการเข้าถึงได้ ตัววัดความเชื่อถือได้ และตัววัดสำหรับความมั่นคง โดยจะเพิ่มฟังก์ชันการทำงานใน WebServiceEndpointStatsProvider ซึ่งเป็นส่วนต่อประสาน (Interface) ของ WebServiceEndpointStatsProviderImpl เพื่อให้ WebServiceEndPointAggregateStats ได้นำไปใช้ต่อไป ดังตารางที่ 4.1

ตารางที่ 4.1 ฟังก์ชันการทำงานใน WebServiceEndpointStatsProvider

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
1. getTotalAvailableTime	เป็นฟังก์ชันที่ใช้เรียกข้อมูลของเวลาทั้งหมดที่เว็บเซอร์วิซมีสภาพพร้อมใช้งาน ที่เก็บรวบรวมโดย WebServiceHealthCheck
2. getTotalMonitoringTime	เป็นฟังก์ชันที่ใช้เรียกข้อมูลของเวลาทั้งหมดที่ใช้ในการเฝ้าสังเกตสภาพพร้อมใช้งาน ที่เก็บรวบรวมโดย WebServiceHealthCheck
3. getACResponseCount	เป็นฟังก์ชันที่ใช้เรียกข้อมูลจำนวนคำร้องขอทั้งหมดที่ทำงานสำเร็จและภายในกรอบเวลาที่คาดหวังที่กำหนดไว้ในแอดมินคอนโซลตั้งแต่ฟังก์ชัน startCollectACMetric ถูกเรียก จนกระทั่งฟังก์ชัน stopCollectACMetric ถูกเรียก

ตารางที่ 4.1 ฟังก์ชันการทำงานใน WebServiceEndpointStatsProvider (ต่อ)

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
4. getReliabilityResponse	เป็นฟังก์ชันที่ใช้เรียกข้อมูลจำนวนคำร้องขอทั้งหมดที่ทำงานสำเร็จและภายในกรอบเวลาที่คาดหวังที่กำหนดไว้ในแอดมินคอนโซลตั้งแต่เริ่มทำการเฝ้าสังเกต
5. getUnReliabilityResponse	เป็นฟังก์ชันที่ใช้เรียกดูข้อมูลจำนวนคำร้องขอทั้งหมดที่ทำงานไม่สำเร็จหรือไม่อยู่ภายในกรอบเวลาที่คาดหวังที่กำหนดไว้ในแอดมินคอนโซลตั้งแต่เริ่มทำการเฝ้าสังเกต
6. getTotalCVSSScore	เป็นฟังก์ชันที่ใช้เรียกดูข้อมูลผลรวมของคะแนนซีวีเอสเอสของรายการซีวีอีทั้งหมดที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ ซึ่งได้มาจากมอดูลวิเคราะห์ความมั่นคง
7. getTotalCVEs	เป็นฟังก์ชันที่ใช้เรียกดูข้อมูลรายการซีวีอีทั้งหมดที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ ซึ่งได้มาจากมอดูลวิเคราะห์ความมั่นคง

ในการเก็บข้อมูลเพื่อคำนวณตัววัดสภาพพร้อมใช้งาน จะมีการสร้างคลาส WebServiceHealthCheck ให้รันเป็นเหมือน เทร็ด (Thread) เพื่อส่งคำร้องขอไปที่เว็บเซอร์วิซแต่ละตัวเป็นช่วง ๆ เพื่อเช็คว่าเว็บเซอร์วิซแต่ละตัวยังมีสภาพที่พร้อมจะให้บริการได้หรือไม่ และทำการเก็บข้อมูลระยะเวลาที่เว็บเซอร์วิซแต่ละตัวมีสภาพที่พร้อมใช้งานและช่วงเวลาในการเฝ้าสังเกตเว็บเซอร์วิซแต่ละตัว เพื่อส่งให้คลาส WebServiceEndpointStatsProviderImpl ต่อไป

ในการเก็บข้อมูลเพื่อคำนวณตัววัดความสามารถในการเข้าถึงได้ จะเก็บจำนวนคำร้องขอที่ทำงานได้สำเร็จและภายใต้กรอบเวลาที่คาดหวังที่กำหนดไว้ในแอดมินคอนโซล เนื่องจากตัววัดความสามารถในการเข้าถึงมีจุดมุ่งหมายเพื่อให้ผู้บริการเว็บเซอร์วิซทดสอบการรับภาระงานในกรณีที่มีการส่งจำนวนคำร้องขอจำนวนมาก ๆ ไปยังเว็บเซอร์วิซและเฝ้าสังเกตจำนวนคำร้องขอที่ทำงานได้สำเร็จและภายใต้กรอบเวลาคาดหวัง การเฝ้าสังเกตเพื่อคำนวณตัววัดชนิดนี้จึงเกิดขึ้นเฉพาะช่วงเวลาใดเวลาหนึ่งเท่านั้น ดังนั้นในการพัฒนาจะเริ่มเก็บข้อมูลจำนวนคำร้องขอก็ต่อเมื่อฟังก์ชัน startCollectACMetric ถูกเรียกหรือกล่าวอีกนัยหนึ่งคือมีการกดปุ่มเริ่มต้นทดสอบการรับภาระงานที่แอดมินคอนโซล และจะหยุดเก็บข้อมูลจำนวนคำร้องขอก็ต่อเมื่อฟังก์ชัน

stopCollectACMetric ถูกเรียกหรือกล่าวอีกนัยหนึ่งคือมีการกดปุ่มเสร็จสิ้นการรับภาระงานที่แอดมินคอนโซล

ในการเก็บข้อมูลตัววัดความน่าเชื่อถือได้ จะเก็บจำนวนคำร้องขอที่ทำงานได้สำเร็จและภายใต้กรอบเวลาที่คาดหวังที่กำหนดไว้ในแอดมินคอนโซลและจำนวนคำร้องขอทั้งหมด ส่วนการเก็บข้อมูลตัววัดความมั่นคง จะเก็บข้อมูลโดยใช้มอดูลวิเคราะห์ความมั่นคงซึ่งจะกล่าวถึงต่อไปในหัวข้อที่ 4.2

4.2 การพัฒนาส่วนของมอดูลวิเคราะห์ความมั่นคง

การทำงานของมอดูลวิเคราะห์ความมั่นคง จะเริ่มทำงานเมื่อจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์เริ่มทำงาน โดยจะทำงานเป็นโพรเซสพื้นหลัง (Background Process) เพื่อไม่ให้กระทบการทำงานของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ โดยจะดึงรายการซอฟต์แวร์ที่อยู่ในเครื่องที่เว็บเซอร์วิซทำงานอยู่ มาจับคู่กับรายการซีวีอีที่มีชื่อซอฟต์แวร์และเลขเวอร์ชันตรงกัน ที่ดึงมาจากส่วนเก็บข้อมูลเอ็นวีดีภายใน และมีฟังก์ชันการทำงานที่ให้ผู้ดูแลระบบสามารถที่จะปรับปรุงรายการซีวีอีในส่วนจัดเก็บข้อมูลเอ็นวีดีภายในโดยใช้ข้อมูลรายการซีวีอีล่าสุดจากเว็บไซต์ของเอ็นวีดี โดยได้สร้างคลาส SecurityAnalyzer ซึ่งมีฟังก์ชันการทำงานดังตารางที่ 4.2

ตารางที่ 4.2 ฟังก์ชันการทำงานในคลาส SecurityAnalyzer

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
1. getTotalCVSSScore	เป็นฟังก์ชันที่ใช้เรียกดูข้อมูลผลรวมของคะแนนซีวีเอสเอสของรายการซีวีอีทั้งหมดที่มีชื่อของซอฟต์แวร์และเลขเวอร์ชันที่มีจุดอ่อนตรงกันกับชื่อของซอฟต์แวร์และเลขเวอร์ชันที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ ซึ่งข้อมูลที่ได้นี้จะนำไปใช้ในการคำนวณค่าตัววัดความมั่นคงต่อไป
2. getTotalCVEs	เป็นฟังก์ชันที่ใช้เรียกดูข้อมูลจำนวนของรายการซีวีอีทั้งหมดที่มีชื่อของซอฟต์แวร์และเลขเวอร์ชันที่มีจุดอ่อนตรงกันกับชื่อของซอฟต์แวร์และเลขเวอร์ชันที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ

ตารางที่ 4.2 ฟังก์ชันการทำงานในคลาส SecurityAnalyzer (ต่อ)

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
3. getRelevantCVEs	เป็นฟังก์ชันที่ใช้เรียกดูรายการกลุ่มของข้อมูลที่ประกอบไปด้วย - ชื่อของซอฟต์แวร์ - เลขเวอร์ชันของซอฟต์แวร์ - รายการซีวีอีทั้งหมดที่มีชื่อและเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อนตรงกันกับชื่อของซอฟต์แวร์และเลขเวอร์ชันของซอฟต์แวร์นี้
4. updateRelevantCVEsWithOnlineNVD	เป็นฟังก์ชันที่เมื่อถูกเรียกจะทำงานดังต่อไปนี้ - ปรับปรุงรายการซีวีอีในส่วนจัดเก็บข้อมูลภายในโดยใช้ข้อมูลล่าสุดจากเว็บไซต์เอ็นวีดี - จับคู่รายการซีวีอีที่ปรับปรุงแล้วกับรายการซอฟต์แวร์ในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ - คำนวณค่าผลรวมของคะแนนซีวีเอสเอสของรายการซีวีอีทั้งหมดหลังปรับปรุงรายการซีวีอี

4.2.1 การดึงรายชื่อซอฟต์แวร์และเลขเวอร์ชันที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ

เนื่องจากจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ เวอร์ชัน 2 ยูอาร์ 1 ถูกติดตั้งในระบบปฏิบัติการวินโดวส์ ดังนั้นในการพัฒนาจะเน้นไปในการดึงรายชื่อซอฟต์แวร์และเลขเวอร์ชันที่พบในระบบปฏิบัติการวินโดวส์ ซึ่งสามารถดึงได้จากวินโดวส์รีจิสทรี (Windows Registry) โดยชื่อและเลขเวอร์ชันของซอฟต์แวร์ทั้งหมดที่ติดตั้งในระบบปฏิบัติการวินโดวส์จะเก็บไว้ใน HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall

ซึ่งรายละเอียดของข้อมูลที่เก็บไว้ในวินโดวส์รีจิสทรี สามารถแสดงได้ดังรูปที่ 4.1

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{AC76BA86-7AD7-1033-7B44-A81200000003}

Comments REG_SZ

Contact REG_SZ Customer Support

DisplayVersion REG_SZ 8.1.2

HelpLink REG_EXPAND_SZ http://www.adobe.com/support/main.html

HelpTelephone REG_SZ

InstallDate REG_SZ 20080210

InstallLocation REG_SZ

InstallSource REG_SZ C:\Users\Tao\AppData\Local\Adobe\Updater5\Install\reader8rdr-en_US\

Publisher REG_SZ Adobe Systems Incorporated

Readme REG_EXPAND_SZ [INSTALLDIR]Reader\Readme.htm

Size REG_SZ

EstimatedSize REG_DWORD 0x15378

UninstallString REG_EXPAND_SZ MsiExec.exe /I{AC76BA86-7AD7-1033-7B44-A81200000003}

DisplayName REG_SZ Adobe Reader 8.1.2

รูปที่ 4.1 รายละเอียดของข้อมูลที่เก็บไว้ในวินโดวส์รีจิสทรี

โดยชื่อของซอฟต์แวร์จะดึงมาจาก "DisplayName" และเลขเวอร์ชันจะดึงมาจาก "DisplayVersion" ซึ่งในส่วนของรายชื่อของซอฟต์แวร์จะกรองเอาเฉพาะชื่อของซอฟต์แวร์ที่ไม่มีคำว่า “Update” และ ไม่มีข้อมูลเกี่ยวกับ "ParentDisplayName" ปรากฏอยู่ด้วย เพราะไม่ได้เป็นชื่อของซอฟต์แวร์ที่แท้จริง เป็นเพียงส่วนเสริมเพิ่มเติมของซอฟต์แวร์ที่ติดตั้งอยู่ในเครื่องเท่านั้น

4.2.2 การจับคู่รายการซีวีอีกับรายการซอฟต์แวร์ในเครื่องที่ให้บริการเว็บเซอร์วิซ

ในแต่ละรายการซีวีอีที่ได้มาจากเว็บไซต์ของเอ็นวีดี จะถูกเก็บรวบรวมเป็นไฟล์ข้อมูลแยกตามปีที่พบรายการซีวีอีนั้น ๆ โดยจะเก็บรวบรวมในรูปแบบของภาษาเอ็กซ์เอ็มแอล ซึ่งในแต่ละรายการซีวีอีจะประกอบด้วย ชื่อและรายละเอียดของซีวีอีรวมถึงวิธีในการแก้ไข วันเดือนปีที่เผยแพร่ซีวีอี (Issue Date) คะแนนซีวีเอสเอส รายชื่อของซอฟต์แวร์และเวอร์ชันของซอฟต์แวร์ที่มี

จุดอ่อนชนิดนี้อยู่ และแหล่งรายการอ้างอิงของซีวีอีนั้น ๆ ซึ่งโครงสร้างของข้อมูลซีวีอีแต่ละตัวในส่วนจัดเก็บข้อมูลเอ็นวีดีสามารถแสดงได้ดังรูปที่ 4.2

รูปที่ 4.2 โครงสร้างของข้อมูลซีวีอีแต่ละตัวในส่วนจัดเก็บข้อมูลเอ็นวีดี

โดยในการจับคู่ระหว่างรายการซีวีอีทั้งหมดที่มีชื่อและเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อนตรงกันกับชื่อของซอฟต์แวร์และเลขเวอร์ชันของซอฟต์แวร์ที่ติดตั้งอยู่เครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซจะใช้หลักการดังต่อไปนี้

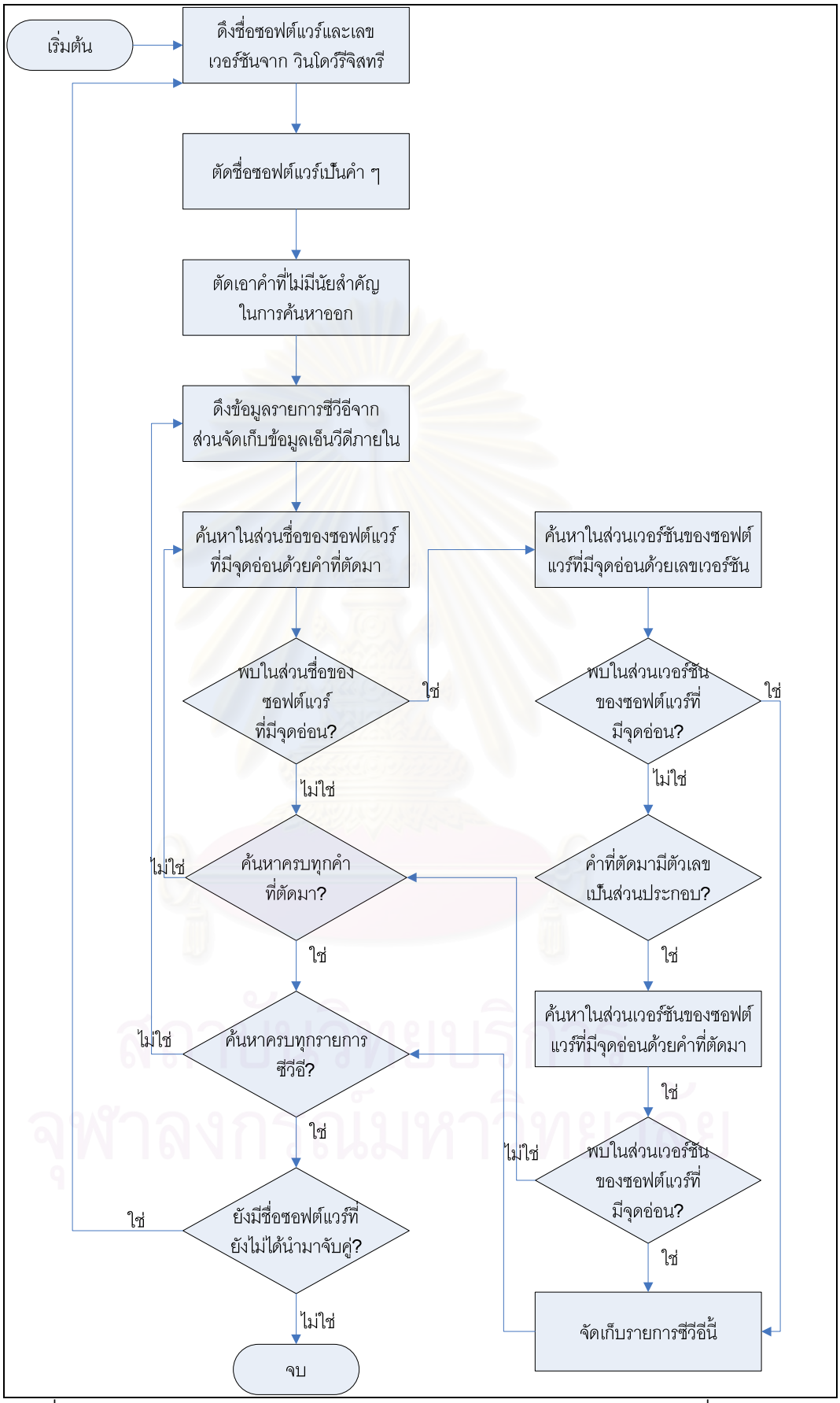
1. แบ่งชื่อของซอฟต์แวร์ที่ได้มาจากวินโดวส์รีจิสทรี ออกเป็นคำ ๆ โดยใช้ช่องว่างระหว่างคำเป็นตัวแบ่ง เนื่องจากชื่อของซอฟต์แวร์ไม่มีมาตรฐานสากลในการตั้งชื่อ ดังนั้นชื่อของซอฟต์แวร์ที่อยู่ในวินโดวส์รีจิสทรีอาจจะมีความแตกต่างกันกับชื่อของซอฟต์แวร์ที่มาจากเว็บไซต์ของเอ็นวีดี การแบ่งชื่อของซอฟต์แวร์เป็นคำ ๆ จะช่วยเพิ่มโอกาสในการจับคู่ได้มากขึ้น
2. ตัดคำที่ไม่มีนัยสำคัญในการค้นหา หรือเป็นคำที่มักปรากฏอยู่ในชื่อของซอฟต์แวร์หลาย ๆ ซอฟต์แวร์ซึ่งถ้านำมาใช้ในการค้นหาอาจเกิดความผิดพลาดได้ ออกจากกลุ่มของคำที่จะนำไปค้นหา เช่น “a”, “the”, “player” เป็นต้น เพื่อเป็นการลดจำนวนคำที่จะนำไปค้นหา ลดความผิดพลาดในการจับคู่ที่อาจเกิดขึ้น และเป็นการลดเวลาในการค้นหาด้วย
3. เอาชื่อของซอฟต์แวร์ที่ผ่านขั้นตอนที่ 1 และ 2 มาจับคู่กับชื่อของซอฟต์แวร์ที่มีจุดอ่อนในแต่ละรายการซีวีอีซึ่งได้มาจากส่วนจัดเก็บข้อมูลเอ็นวีดีภายใน โดยแบ่งชื่อของซอฟต์แวร์ที่มีจุดอ่อนออกเป็นคำ ๆ โดยใช้ช่องว่างระหว่างคำเป็นตัวแบ่งเช่นกัน แล้วนำมาจับคู่กับชื่อของซอฟต์แวร์ที่ได้มาจากวินโดวส์รีจิสทรี
4. ถ้าได้รายการซีวีอีที่มีชื่อของซอฟต์แวร์ที่มีจุดอ่อนตรงกับคำที่นำมาค้นหา ให้เอาเลขเวอร์ชันของซอฟต์แวร์มาเปรียบเทียบว่าตรงกันกับเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อนในรายการซีวีอี

นั้น ๆ หรือไม่ ถ้าตรงกันจะทำการจัดเก็บรายการซีวีอีนั้นลงในรายการซีวีอีที่เป็นผลกระทบมาจากการติดตั้งซอฟต์แวร์ตัวนี้

5. ในกรณีที่เลขเวอร์ชันของซอฟต์แวร์ไม่ตรงกันกับเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อนในรายการซีวีอีนั้น ๆ ให้กลับไปใช้ชื่อของซอฟต์แวร์ที่ถูกแบ่งเป็นคำ ๆ ในขั้นตอนที่ 1 และ 2 มาใช้ในการเปรียบเทียบว่าตรงกันกับเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อนในรายการซีวีอีนั้น ๆ หรือไม่ โดยจะเลือกเฉพาะคำที่มีตัวเลขเป็นส่วนประกอบด้วย ซึ่งถ้าคำ ๆ นั้นตรงกันกับเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อนในรายการซีวีอีนั้น ๆ จะทำการจัดเก็บรายการซีวีอีนั้นลงในรายการซีวีอีที่เป็นผลกระทบมาจากการติดตั้งซอฟต์แวร์ตัวนี้

การทำงานของโปรแกรมจับคู่รายการซีวีอีกับรายการซอฟต์แวร์ในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซสามารถแสดงได้ดังรูปที่ 4.3

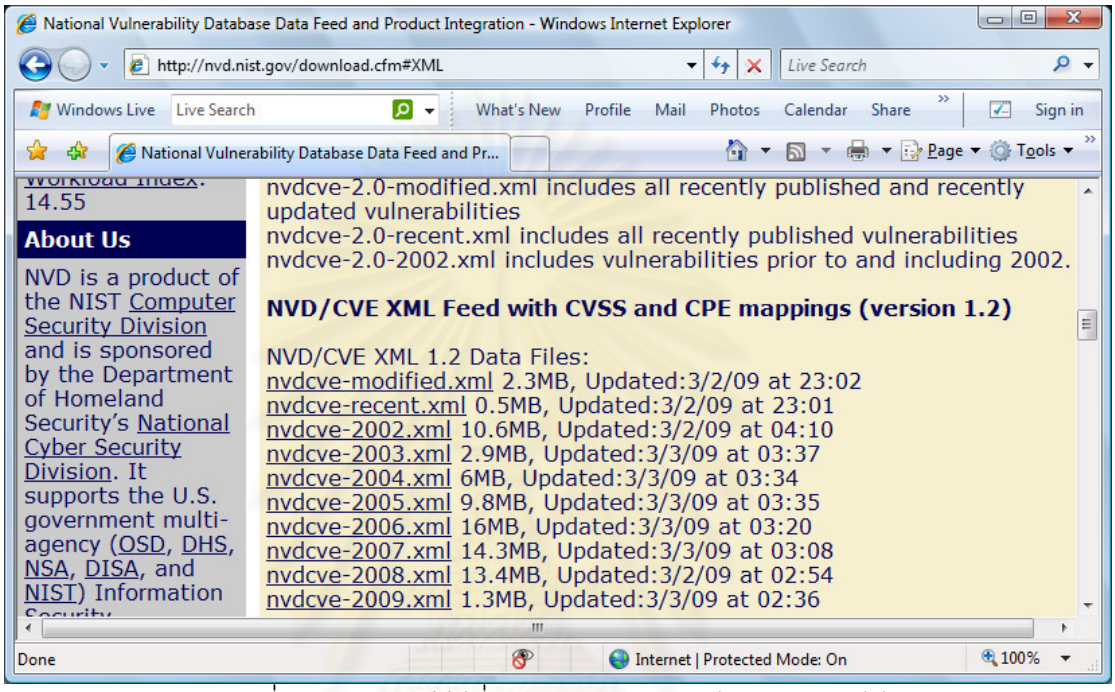
สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย



รูปที่ 4.3 การทำงานของโปรแกรมจับคู่รายการซีวีอีกับรายการซอฟต์แวร์ในเครื่องเซิร์ฟเวอร์

4.2.3 การปรับปรุงข้อมูลรายการซีวีอีในส่วนจัดเก็บข้อมูลเอ็นวีดีภายใน

จากที่กล่าวในหัวข้อที่ 4.2.2 รายการซีวีอีทั้งหมดในเว็บไซต์เอ็นวีดีจะถูกเก็บรวบรวมเป็นไฟล์ข้อมูลแยกตามปีที่พบรายการซีวีอีนั้น ๆ โดยรายการซีวีอีทั้งหมดจะถูกเผยแพร่ผ่านทางเว็บไซต์ของเอ็นวีดีดังรูปที่ 4.4



รูปที่ 4.4 รายการซีวีอีที่เผยแพร่ผ่านทางเว็บไซต์ของเอ็นวีดี

เว็บไซต์ของเอ็นวีดีจะมีการปรับปรุงข้อมูลรายการซีวีอีทุกวัน ดังนั้นจึงมีการสร้างฟังก์ชัน updateRelevantCVEsWithOnlineNVD เพื่อให้มอดูลวิเคราะห์ความมั่นคงสามารถปรับปรุงข้อมูลในส่วนจัดเก็บข้อมูลภายในโดยการดาวน์โหลดข้อมูลรายการซีวีอีทั้งหมดจากเว็บไซต์เอ็นวีดีผ่านทางเอชทีพีพีโพรโทคอล (HTTP Protocol) และทำการจับคู่รายการซีวีอีและคำนวณค่าตัววัดความมั่นคงใหม่อีกครั้งหนึ่งโดยใช้ข้อมูลล่าสุดจากเว็บไซต์เอ็นวีดี โดยชื่อของไฟล์ข้อมูลรายการซีวีอีทั้งหมดและตำแหน่งที่จัดเก็บจะถูกเก็บไว้ในไฟล์ ws-metrics-config.xml เพื่อให้ง่ายในการแก้ไขเปลี่ยนแปลงผ่านทางแอดมินคอนโซล ดังรูปที่ 4.5

```
<Vulnerability_Info_Files
location="D:\workspace\publish\glassfish\domains\domain1\config\Vulnerability_Info_Files\">
        <name>nvdcve-2002.xml</name>
        <name>nvdcve-2003.xml</name>
        <name>nvdcve-2004.xml</name>
        <name>nvdcve-2005.xml</name>
        <name>nvdcve-2006.xml</name>
        <name>nvdcve-2007.xml</name>
        <name>nvdcve-2008.xml</name>
        <name>nvdcve-2009.xml</name>
</Vulnerability_Info_Files
```

รูปที่ 4.5 ชื่อของไฟล์ข้อมูลรายการซีวีอีและตำแหน่งที่จัดเก็บในไฟล์ ws-metrics-config.xml

4.3 การพัฒนาส่วนของมอดูลในการคำนวณและจัดเก็บค่าตัววัด

4.3.1 ส่วนขยายของเว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตต (Extended Web Service EndPoint AggregateStats)

เว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตต (WebServiceEndPointAggregateStats) เป็นส่วนติดต่อกับผู้ใช้ที่อนุญาตให้ผู้ใช้สามารถเข้าถึงข้อมูลในแต่ละเมเนจบีนได้ โดยจะดึงข้อมูลจากเว็บเซอร์วิซเอนด์พอยนต์สแตตโพรไวเดอร์ (WebServiceEndpointStatsProvider) มาอีกต่อหนึ่ง มอดูลในการคำนวณจะสามารถเข้าถึงค่าข้อมูลในแต่ละเมเนจบีนผ่านทาง เว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตต เพื่อนำไปใช้ในการคำนวณค่าตัววัดต่าง ๆ โดยจะมีการเพิ่มฟังก์ชันลงในเว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตต เพื่อนำไปใช้ในการคำนวณค่าตัววัดต่าง ๆ ซึ่งฟังก์ชันที่ให้ข้อมูลในการคำนวณค่าตัววัดของเว็บเซอร์วิซเอนด์พอยนต์แอกกรีเกตสแตต สามารถแสดงได้ดังตารางที่ 4.3

ตารางที่ 4.3 ฟังก์ชันที่ให้ข้อมูลในการคำนวณค่าตัววัดของ WebServiceEndPointAggregateStats

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
1. getAverageResponseTime	เป็นฟังก์ชันที่ให้ข้อมูลเวลาในการตอบสนองกลับโดยเฉลี่ยของเว็บเซอร์วิซแต่ละตัว ซึ่งข้อมูลนี้จะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับสมรรถนะ

ตารางที่ 4.3 ฟังก์ชันที่ให้ข้อมูลในการคำนวณค่าตัววัดของ WebServiceEndPointAggregateStats (ต่อ)

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
2. getTotalAvailableTime	เป็นฟังก์ชันที่ให้ข้อมูลของเวลาทั้งหมดที่เว็บเซอร์วิซมีสภาพพร้อมใช้งาน ซึ่งข้อมูลนี้จะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับสภาพพร้อมใช้งาน
3. getTotalMonitoringTime	เป็นฟังก์ชันที่ให้ข้อมูลของเวลาทั้งหมดที่ใช้ในการเฝ้าสังเกตสภาพพร้อมใช้งาน ซึ่งจะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับสภาพพร้อมใช้งาน
4. getACResponseCount	เป็นฟังก์ชันที่ให้ข้อมูลจำนวนคำร้องขอทั้งหมดที่ทำงานสำเร็จและภายในกรอบเวลาที่คาดหวังที่กำหนดไว้ในแอดมินคอนโซล ซึ่งข้อมูลนี้จะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับความสามารถในการเข้าถึงได้
5. getReliabilityResponse	เป็นฟังก์ชันที่ให้ข้อมูลจำนวนคำร้องขอทั้งหมดที่ทำงานสำเร็จและภายในกรอบเวลาที่คาดหวังที่กำหนดไว้ในแอดมินคอนโซลตั้งแต่เริ่มทำการเฝ้าสังเกต ซึ่งข้อมูลนี้จะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับความเชื่อถือได้
6. getUnReliabilityResponse	เป็นฟังก์ชันที่ให้ข้อมูลจำนวนคำร้องขอทั้งหมดที่ทำงานไม่สำเร็จหรือไม่อยู่ภายในกรอบเวลาที่คาดหวังที่กำหนดไว้ในแอดมินคอนโซลตั้งแต่เริ่มทำการเฝ้าสังเกต ซึ่งข้อมูลนี้จะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับความเชื่อถือได้
7. getTotalCVSSScore	เป็นฟังก์ชันที่ให้ข้อมูลผลรวมของคะแนนซีวีเอสเอสของรายการซีวีอีทั้งหมดที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ ซึ่งได้มาจากมอดูลวิเคราะห์ความมั่นคง ซึ่งข้อมูลนี้จะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับความมั่นคง
8. getTotalCVEs	เป็นฟังก์ชันที่ให้ข้อมูลรายการซีวีอีทั้งหมดที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ ซึ่งได้มาจากมอดูลวิเคราะห์ความมั่นคง ซึ่งข้อมูลนี้จะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับความมั่นคง

4.3.2 มอดูลในการคำนวณค่าตัววัด

มอดูลในการคำนวณค่าตัววัดจะประกอบด้วย 2 ส่วนหลัก ๆ ซึ่งได้แก่

- WebServiceEndpointAggregateMetrics เป็นส่วนต่อประสาน (Interface) ที่จะอนุญาตให้ผู้ใช้สามารถเข้าถึงค่าของตัววัดที่ได้จากการคำนวณ โดยจะถูกเรียกใช้โดย WebServiceHandlers เพื่อนำค่าตัววัดไปแสดงผลที่แอดมินคอนโซล โดยฟังก์ชันทั้งหมดที่สามารถเรียกใช้ผ่าน WebServiceEndpointAggregateMetrics สามารถแสดงได้ดังตารางที่ 4.4

ตารางที่ 4.4 ฟังก์ชันทั้งหมดที่สามารถเรียกใช้ผ่าน WebServiceEndpointAggregateMetrics

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
1. getAvailabilityMetric	เป็นฟังก์ชันที่ให้ข้อมูลค่าตัววัดสำหรับสภาพพร้อมใช้งานที่ได้จากการคำนวณ
2. getAccessibilityMetric	เป็นฟังก์ชันที่ให้ข้อมูลค่าตัววัดสำหรับความสามารถในการเข้าถึงได้ที่ได้จากการคำนวณ
3. getPerformanceMetric	เป็นฟังก์ชันที่ให้ข้อมูลค่าตัววัดสำหรับสมรรถนะ
4. getReliabilityMetric	เป็นฟังก์ชันที่ให้ข้อมูลค่าตัววัดสำหรับความเชื่อถือได้ที่ได้จากการคำนวณ
5. getSecurityMetric	เป็นฟังก์ชันที่ให้ข้อมูลค่าตัววัดสำหรับความมั่นคงที่ได้จากการคำนวณ
6. getRegulatoryMetric	เป็นฟังก์ชันที่ให้ข้อมูลค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบที่ได้จากการคำนวณ

- WebServiceEndpointAggregateMetricsImpl เป็นคลาสที่ทำการคำนวณค่าตัววัดต่าง ๆ โดยอิมพลีเมนต์อินเทอร์เฟซ WebServiceEndpointAggregateMetrics เพื่อให้ WebServiceHandlers สามารถเข้าถึงผลลัพธ์ที่ได้จากการคำนวณได้ ซึ่งจะรับอ็อบเจกต์ WebServiceEndPointAggregateStats และชื่อของเว็บเซอร์วิซเป็นพารามิเตอร์เพื่อใช้ในการคำนวณค่าตัววัดต่าง ๆ และจะใช้สูตรการคำนวณในหัวข้อที่ 3.1 ซึ่งรายละเอียดในแต่ละฟังก์ชันการคำนวณค่าสามารถแสดงได้ดังตารางที่ 4.5

ตารางที่ 4.5 ฟังก์ชันการคำนวณค่าของ WebServiceEndpointAggregateMetrics

ชื่อฟังก์ชัน	รายละเอียดในการทำงาน
1. getAvailabilityMetric	● ดึงค่าเวลาทั้งหมดที่เว็บเซอร์วิซมีสภาพพร้อมใช้งานจากฟังก์ชัน getTotalAvailableTime ● ดึงค่าเวลาทั้งหมดที่ใช้ในการเฝ้าสังเกตสภาพพร้อมใช้งานจากฟังก์ชัน getTotalMonitoringTime ● นำค่าที่ได้ทั้งสองมาหารกันเพื่อจะได้ค่าตัววัดสำหรับสภาพพร้อมใช้งาน
2. getAccessibilityMetric	● ดึงค่าจำนวนคำร้องขอทั้งหมดที่ทำงานสำเร็จและภายในกรอบเวลาที่คาดหวังจากฟังก์ชัน getACResponseCount ● ดึงค่าเวลาทั้งหมดที่ใช้ในการเฝ้าสังเกตที่ได้กำหนดไว้ในแอดมินคอนโซลจากไฟล์ ws-metrics-config.xml ● นำค่าที่ได้ทั้งสองมาหารกันเพื่อจะได้ค่าตัววัดสำหรับความสามารถในการเข้าถึงได้
3. getPerformanceMetric	เนื่องจากค่าตัววัดสำหรับสมรรถนะคำนวณมาจากค่าเฉลี่ยของเวลาที่เว็บเซอร์วิซใช้ในการประมวลผล ซึ่งมีอยู่ในแล้วในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ จึงสามารถนำมาใช้ได้เลย โดยดึงจากฟังก์ชัน getAverageResponseTime
4. getReliabilityMetric	● ดึงค่าจำนวนคำร้องขอทั้งหมดที่ทำงานสำเร็จและภายในกรอบเวลาที่คาดหวังจากฟังก์ชัน getReliabilityResponse ● ดึงค่าจำนวนคำร้องขอทั้งหมดที่ทำงานไม่สำเร็จหรือไม่อยู่ภายในกรอบเวลาที่คาดหวังจากฟังก์ชัน getUnReliabilityResponse ● นำค่าแรกมาหารด้วยผลรวมของค่าที่หนึ่งและค่าที่สองเพื่อจะได้ค่าตัววัดสำหรับความเชื่อถือได้

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ตารางที่ 4.5 ฟังก์ชันการคำนวณค่าของ WebServiceEndpointAggregateMetrics (ต่อ)

ชื่อฟังก์ชัน	รายละเอียดในการทำงาน
5. getSecurityMetric	● ดึงค่าผลรวมของคะแนนซีวีเอสเอสของรายการซีวีอีทั้งหมดที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซจากฟังก์ชัน getTotalCVSSScore ● ดึงค่ารายการซีวีอีทั้งหมดที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซจากฟังก์ชัน getTotalCVEs ● นำค่าที่ได้ทั้งสองมาหารกันเพื่อจะได้ค่าตัววัดสำหรับความมั่นคง
6. getRegulatoryMetric	● ดึงค่าของตัววัดต่างๆที่ได้ประกาศไว้ในแอดมินคอนโซลจากไฟล์ ws-metrics-config.xml ● นำค่าที่ได้มาคำนวณโดยใช้สูตรการคำนวณในหัวข้อที่ 3.1.6 เพื่อจะได้ค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ

4.3.3 **ส่วนจัดเก็บค่าตัววัด** (Metrics History Repository)

เพื่อให้สามารถนำค่าตัววัดที่ได้จากมอดูลในการเฝ้าสังเกต ไปใช้ในการวิเคราะห์เพื่อปรับปรุงคุณภาพในการให้บริการของเว็บเซอร์วิซให้ดีขึ้น จึงต้องมีการเก็บค่าตัววัดที่คำนวณได้เอาไว้ในส่วนจัดเก็บค่าตัววัด ซึ่งเป็นไฟล์ในรูปแบบภาษาเอ็กซ์เอ็มแอล ดังรูปที่ 4.6

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <metrics_history>
  - <metrics wsname="ComputeFactorialService" ipaddress="127.0.0.1" datetime="2009-04-27T11:43:37+07:00">
      <Availability>1.000</Availability>
      <Accessibility>1.008</Accessibility>
      <Performance>829.000</Performance>
      <Reliability>0.834</Reliability>
      <Security>8.544</Security>
      <Regulatory>0.391</Regulatory>
    </metrics>
  - <metrics wsname="ComputeFactorialService" ipaddress="127.0.0.1" datetime="2009-04-27T11:44:37+07:00">
      <Availability>1.000</Availability>
      <Accessibility>1.008</Accessibility>
      <Performance>829.000</Performance>
      <Reliability>0.834</Reliability>
      <Security>8.544</Security>
      <Regulatory>0.391</Regulatory>
    </metrics>
  </metrics_history>
```

รูปที่ 4.6 โครงสร้างในไฟล์เอ็กซ์เอ็มแอลเพื่อใช้ในการจัดเก็บค่าตัววัด

สิ่งที่เก็บในไฟล์จะประกอบด้วยชื่อของเว็บเซอร์วิซ เลขที่อยู่ไอพี(IP Address)ของเว็บเซอร์วิซ เวลาที่จัดเก็บค่าตัววัดต่าง ๆ และค่าของตัววัดต่าง ๆ ที่คำนวณได้ในเวลานั้น โดยจะเก็บค่าตัววัดหนึ่งไฟล์ต่อหนึ่งเว็บเซอร์วิซเพื่อสะดวกต่อการนำไปวิเคราะห์ต่อไปและเพื่อให้มีการจัดเก็บค่าตัววัดต่าง ๆ ของเว็บเซอร์วิซแต่ละตัวในช่วงเวลาที่กำหนดได้ ดังนั้นจะสร้างคลาส MetricsCollector ที่ใช้เก็บข้อมูลให้รันเป็นเหมือนเทร็ด (Thread) เพื่อให้สามารถเก็บค่าตัววัดในทุก ๆ ช่วงเวลาที่เราต้องการ เช่น ทุก ๆ 1 นาที หรือ ทุก ๆ 5 นาทีเป็นต้น โดยตำแหน่งที่เก็บไฟล์เหล่านี้ และช่วงเวลาที่ทำการเก็บค่าตัววัดต่าง ๆ ของเว็บเซอร์วิซแต่ละตัวจะถูกเก็บไว้ในไฟล์ ws-metrics-config.xml ดังรูปที่ 4.7

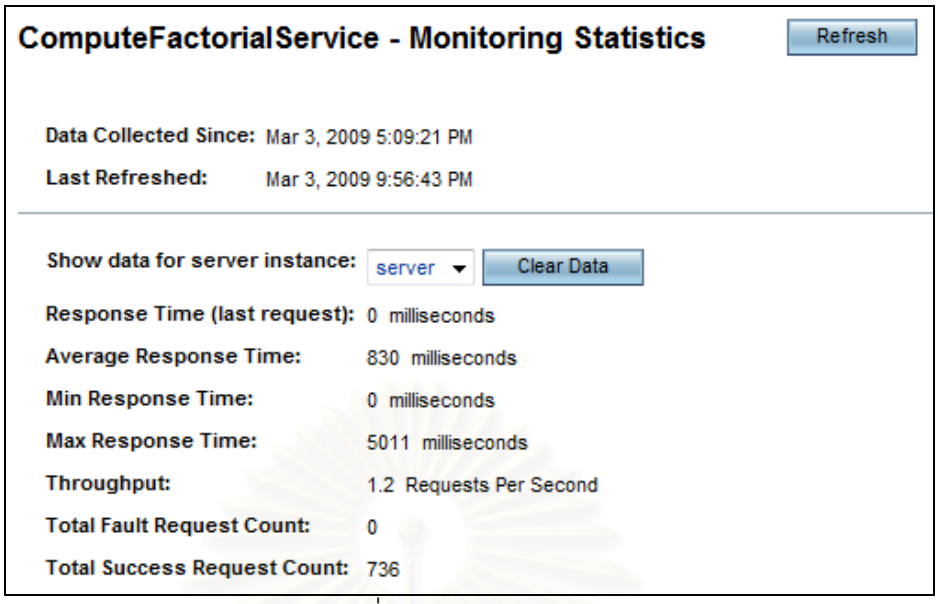
```
- <webservice name="ComputeFactorialService">
  - <metrics>
      <Availability>0.95</Availability>
      <Accessibility>0.80</Accessibility>
      <Performance>4000</Performance>
      <Reliability>0.75</Reliability>
      <Security>8.20</Security>
    </metrics>
  - <loadtest>
      <duration>600</duration>
    </loadtest>
  - <healthcheck>
      <Interval>30</Interval>
      <Timeout>10</Timeout>
    </healthcheck>
  - <metricshistory>
      <Interval>2</Interval>
      <Location>D:\workspace\publish\glassfish\domains\domain1\config\</Location>
    </metricshistory>
  </webservice>
```

รูปที่ 4.7 ความถี่ในการจัดเก็บค่าตัววัดและตำแหน่งที่จัดเก็บในไฟล์ ws-metrics-config

4.4 การพัฒนาส่วนแสดงผลของตัววัดในแอดมินคอนโซล

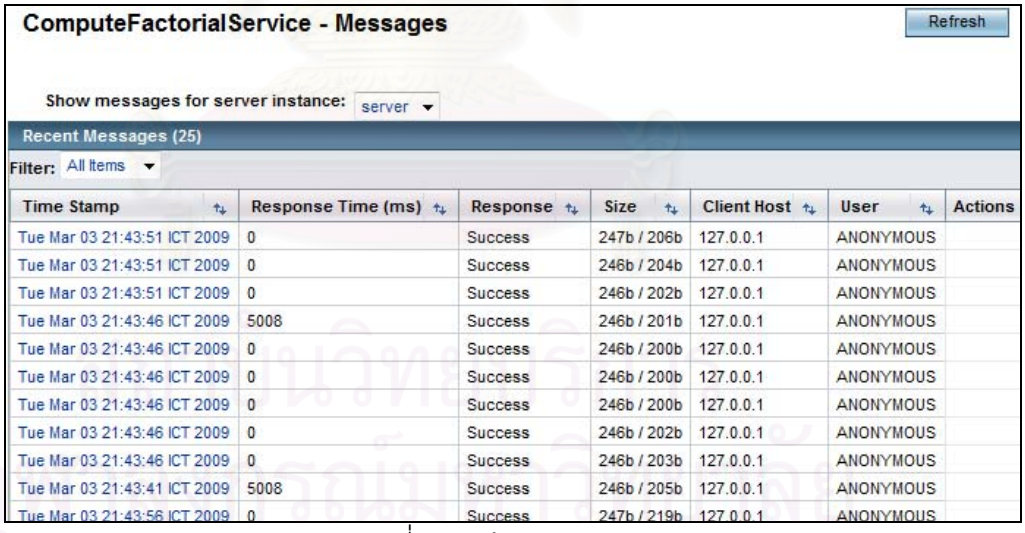
เป็นการพัฒนาในส่วนแสดงผลในแอดมินคอนโซลของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ ซึ่งเป็นเว็บแอพพลิเคชันที่แอดมินสามารถล็อกอินเข้าไปดูรายละเอียดต่าง ๆ ของแอพพลิเคชันเซิรฟเวอร์ ที่ได้ติดตั้งลงไป เช่น จำนวนของเว็บเซอร์วิซหรือจำนวนของเว็บแอพพลิเคชันที่ได้ติดตั้งในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ รายละเอียดในการจัดการหรือควบคุมเว็บแอพพลิเคชันต่าง ๆ เป็นต้น ในส่วนของฟังก์ชันในการเฝ้าสังเกตเว็บเซอร์วิซที่มีอยู่ปัจจุบันในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ประกอบด้วย

- หน้า Statistics เป็นหน้าที่แสดงผลลัพธ์ที่ได้จากการเฝ้าสังเกตเว็บเซอร์วิซในช่วงเวลาตั้งแต่เริ่มเก็บข้อมูลจนถึงปัจจุบัน ซึ่งได้แก่ ระยะเวลาที่เว็บเซอร์วิซตอบกลับโดยเฉลี่ย (Average Response Time) ปริมาณงานหรือจำนวนการร้องขอต่อวินาที (Throughput) เป็นต้น ซึ่งสามารถแสดงได้ดังรูปที่ 4.8



รูปที่ 4.8 หน้า Statistics

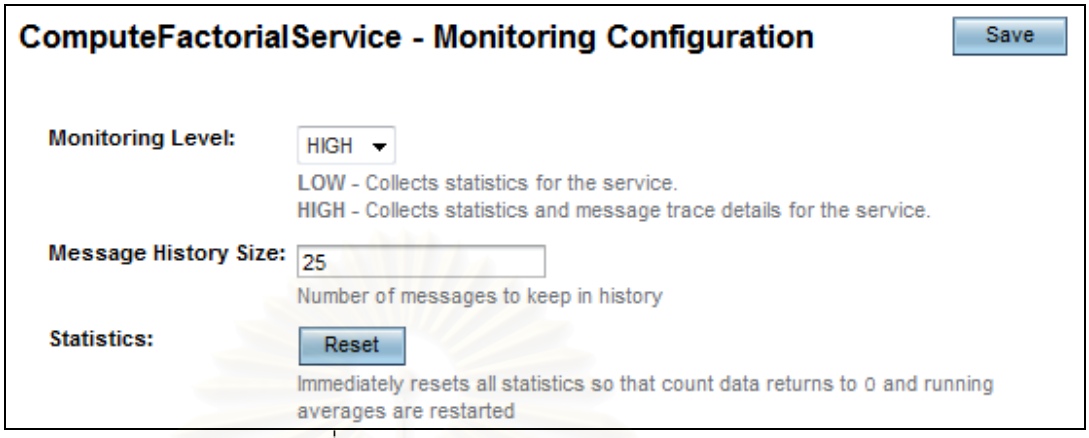
- หน้า Messages เป็นหน้าที่แสดงข้อมูลของรายละเอียดของคำร้องขอและผลลัพธ์ที่ได้ของเว็บเซอร์วิซทั้งหมดตั้งแต่เริ่มเก็บข้อมูลจนถึงปัจจุบัน ในรูปแบบของข้อความโซพ ทั้งในส่วนของหัวเรื่อง (Header) และ เนื้อหา (Body) ของข้อความ ซึ่งสามารถแสดงได้ดังรูปที่ 4.9



รูปที่ 4.9 หน้า Messages

- หน้า Configuration เป็นหน้าที่จัดการเกี่ยวกับพฤติกรรมในการเฝ้าสังเกต อันได้แก่การปรับระดับในการเฝ้าสังเกต การปรับจำนวนของคำร้องขอที่จาวาซิสเต็มแอพพลิเคชันเซิรฟเวอร์จะเก็บไว้ในหน่วยความจำ (Memory) และผู้ดูแลระบบยังสามารถล้างข้อมูล

ในการเฝ้าสังเกตทั้งหมดตั้งแต่เริ่มทำการเฝ้าสังเกตจนถึงปัจจุบัน เพื่อเริ่มทำการเก็บข้อมูลใหม่ได้ ซึ่งสามารถแสดงได้ดังรูปที่ 4.10

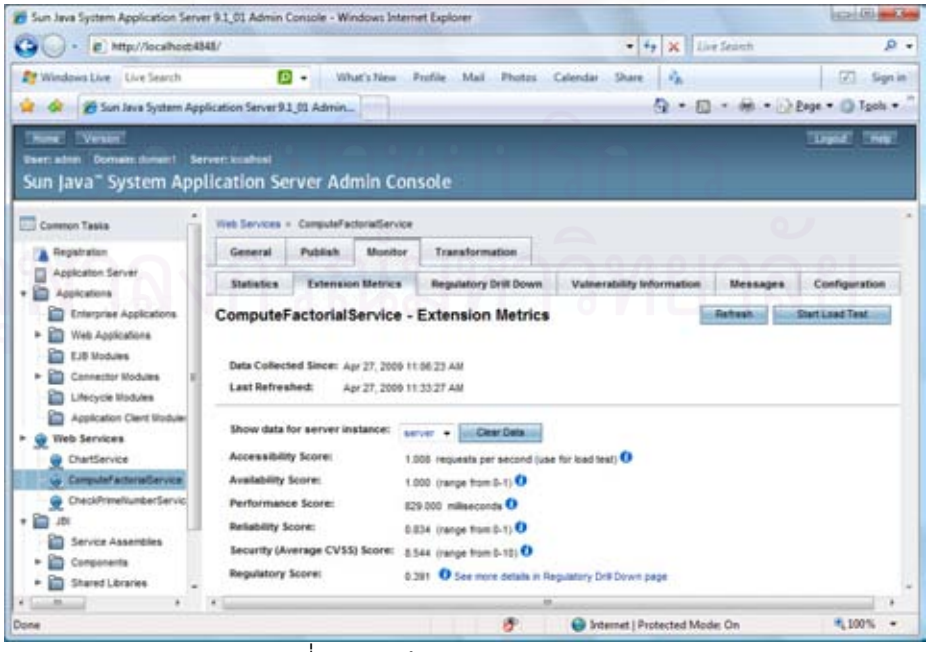


รูปที่ 4.10 หน้า Configuration

ในงานวิจัยนี้จะเพิ่มหน้าเว็บเพื่อแสดงข้อมูลที่ได้มาจากมอดูลในการเฝ้าสังเกตในแอดมินคอนโซล ซึ่งได้แก่ หน้า Extension Metrics และ หน้า Vulnerability Information นอกจากนี้ยังได้เพิ่มรายละเอียดในการบริหารจัดการมอดูลในการเฝ้าสังเกตที่สร้างขึ้นมา ไว้ในหน้า Configuration ซึ่งจะกล่าวถึงในรายละเอียดต่อไป

4.4.1 หน้า Extension Metrics

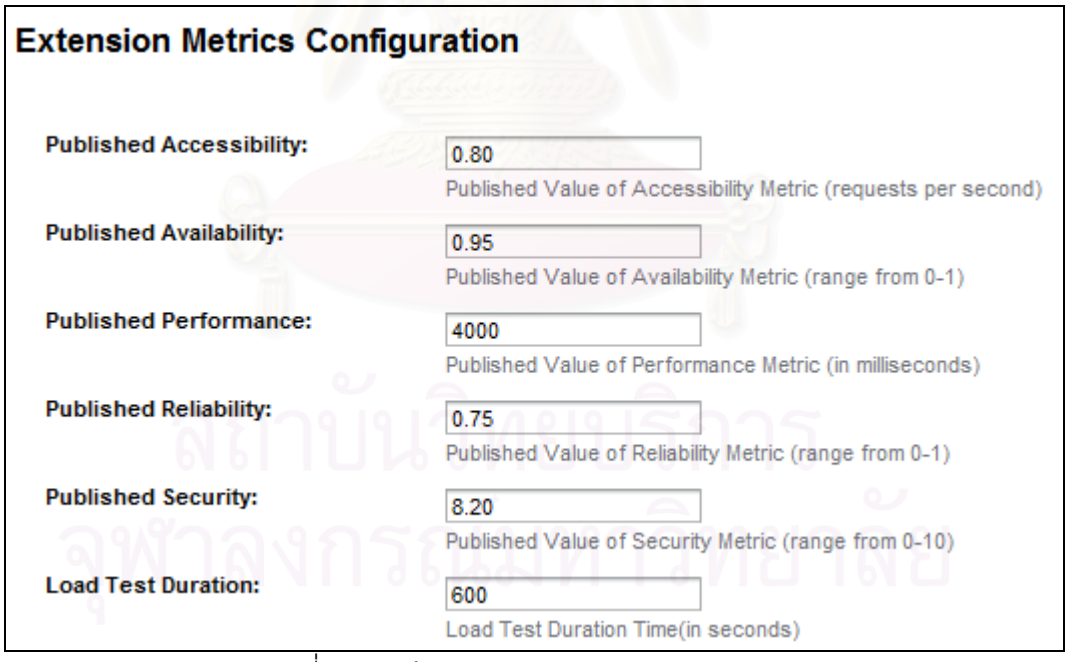
จะแสดงรายละเอียดของค่าตัววัดต่าง ๆ ที่ ได้จากมอดูลการเฝ้าสังเกตตั้งแต่เริ่มต้นทำการเฝ้าสังเกตจนถึงปัจจุบัน ซึ่งรายละเอียดของหน้า Extension Metrics สามารถแสดงได้ดังรูปที่ 4.11



รูปที่ 4.11 หน้า Extension Metrics

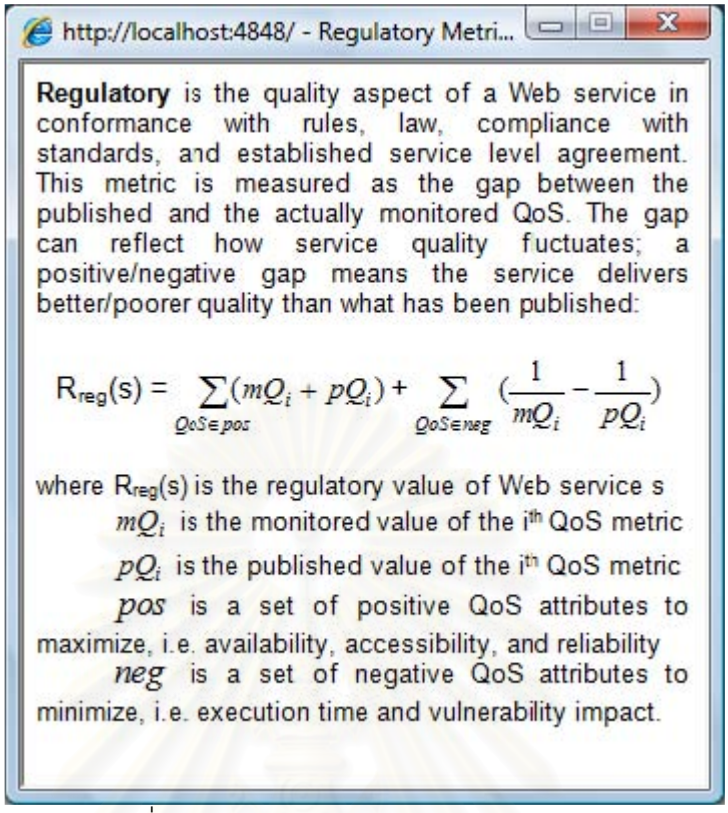
ค่าของตัววัดทุกค่าที่แสดงในหน้านี้ (ยกเว้นค่าความสามารถในการเข้าถึงได้) จะมีค่าเปลี่ยนแปลงไปตามคำร้องขอที่เข้ามาที่เว็บเซอร์วิซนั้น ในส่วนของตัววัดความสามารถในการเข้าถึงได้ ซึ่งมีจุดมุ่งหมายเพื่อให้ผู้บริการเว็บเซอร์วิซทดสอบการรับภาระงานในกรณีที่มีการส่งจำนวนคำร้องขอจำนวนมาก ๆ ไปยังเว็บเซอร์วิซและเฝ้าสังเกตจำนวนคำร้องขอที่ทำงานได้สำเร็จและภายใต้กรอบเวลาคาดหวัง การเฝ้าสังเกตเพื่อคำนวณตัววัดชนิดนี้จึงเกิดขึ้นในช่วงเวลาใดเวลาหนึ่งเท่านั้น ดังนั้นจะเริ่มทำการเฝ้าสังเกตเมื่อมีการกดปุ่ม Start Load Test เท่านั้น และจะหยุดทำการเฝ้าสังเกตเมื่อทำการกดปุ่ม Stop Load Test และจะคำนวณค่าความสามารถในการเข้าถึงได้ออกมา โดยกรอบเวลาที่คาดหวังและเวลาในการเฝ้าสังเกตของตัววัดความสามารถในการเข้าถึงได้จะถูกกำหนดไว้ในหน้า Configuration ดังรูปที่ 4.12

ในส่วนของตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ สามารถที่จะเข้าไปดูรายละเอียดของตัววัดที่ได้จากการเฝ้าสังเกต และตัววัดที่ประกาศไว้ในแอดมินคอนโซลได้จากหน้า Regulatory Drill Down โดยคลิกที่ See more detail in Regulatory Drill Down Page ซึ่งรายละเอียดของหน้า Regulatory Drill Down จะกล่าวถึงต่อไปในหัวข้อที่ 4.4.2



รูปที่ 4.12 หน้า Extension Metrics Configuration

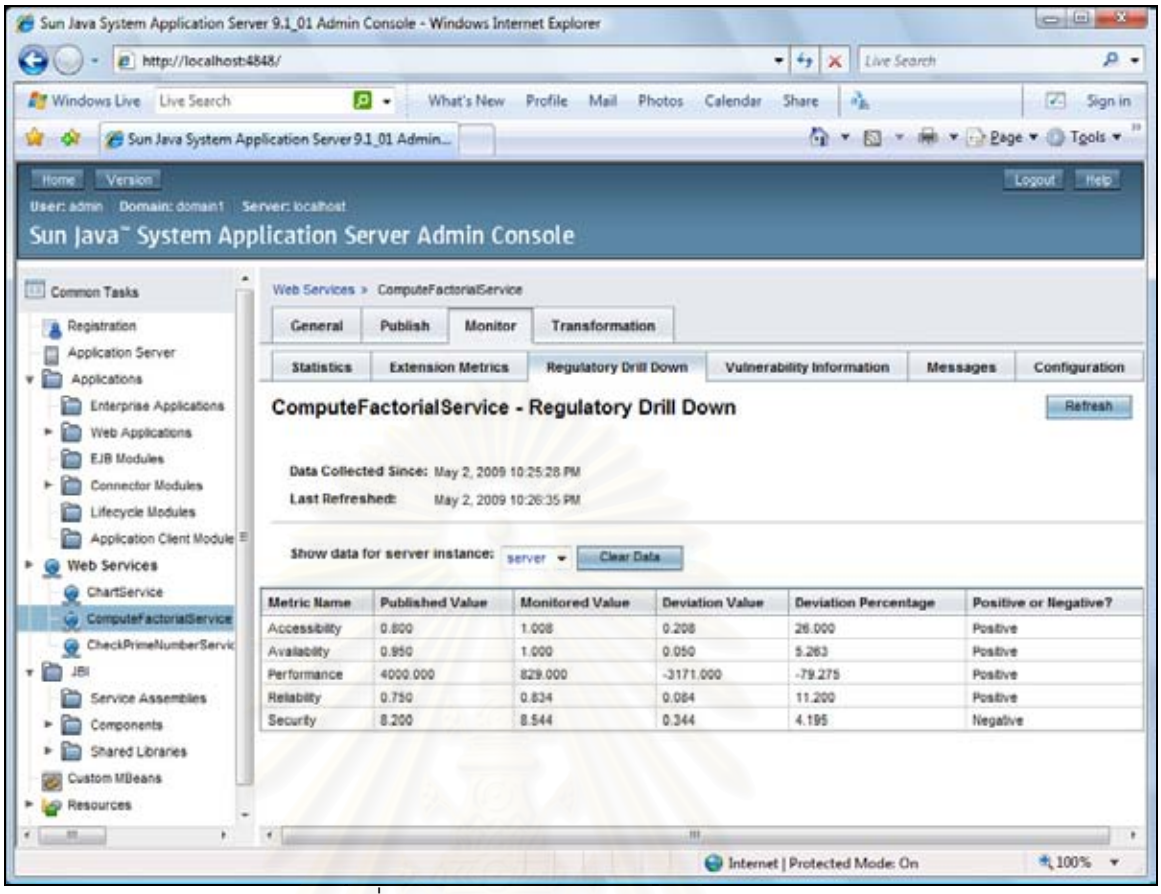
นอกจากนี้แอดมินสามารถล้างค่าตัววัดทั้งหมดที่คำนวณจากมอดูลในการเฝ้าสังเกต และเริ่มทำการเฝ้าสังเกตใหม่โดยใช้ปุ่ม Clear Data และยังดูสูตรในการคำนวณค่าตัววัดแต่ละตัวโดยคลิกที่รูป ℹ ที่อยู่ด้านหลังของค่าตัววัดที่ต้องการดูสูตรในการคำนวณดังรูปที่ 4.13



รูปที่ 4.13 รายละเอียดของสูตรในการคำนวณ

4.4.2 **หน้า** Regulatory Drill Down

จะแสดงรายละเอียดของค่าต่าง ๆ ที่นำมาใช้ในการคำนวณตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ ซึ่งได้แก่ ค่าตัววัดต่าง ๆ ที่ได้ประกาศไว้ในแอดมินคอนโซล (Published Value) และค่าตัววัดที่ได้จากการเฝ้าสังเกตในขณะนั้น (Monitored Value) นอกจากนั้นยังแสดงรายละเอียดของค่าความเบี่ยงเบนที่วัดได้ในตัววัดแต่ละตัวในขณะนั้น (Deviation Value) ซึ่งคำนวณจากผลต่างของค่าตัววัดที่ได้จากการเฝ้าสังเกตกับค่าตัววัดที่ได้ประกาศไว้ ส่วนเปอร์เซ็นต์เบี่ยงเบน (Deviation Percentage) คำนวณได้จากผลต่างของค่าตัววัดที่ได้จากการเฝ้าสังเกตกับค่าตัววัดที่ได้ประกาศไว้แล้วหารด้วยค่าของตัววัดที่ได้ประกาศไว้ ในคอลัมน์สุดท้ายของตารางจะบอกว่าค่าตัววัดที่ได้จากการเฝ้าสังเกตแต่ละชนิด มีการเบี่ยงเบนไปในแง่บวก (Positive) คือเบี่ยงเบนไปในทางที่ดี หรือในแง่ลบ (Negative) คือเบี่ยงเบนไปในทางที่ไม่ดี ซึ่งหน้า Regulatory Drill Down สามารถแสดงได้ดังรูปที่ 4.14

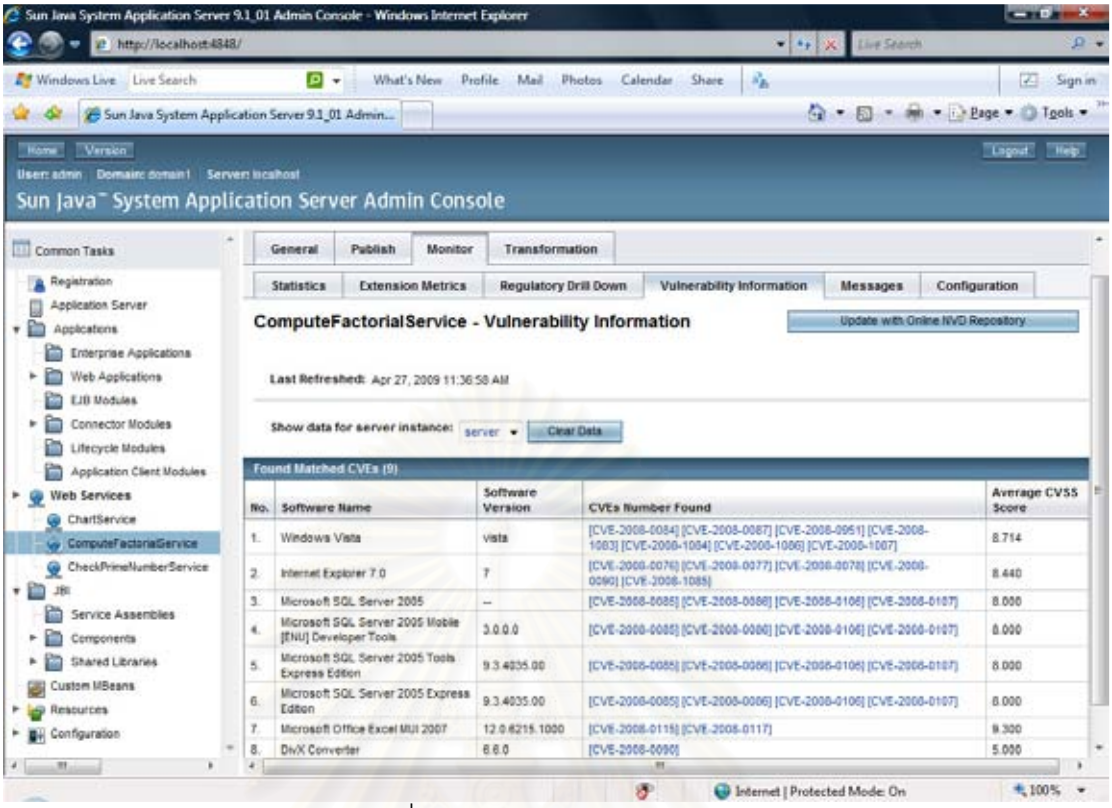


รูปที่ 4.14 หน้า Regulatory Drill Down

จากรูปที่ 4.14 ค่าตัววัดความมั่นคงที่เฝ้าสังเกตได้เป็น 8.544 ซึ่งมีค่ามากกว่าค่าตัววัดความมั่นคงที่ได้ประกาศซึ่งก็คือ 8.200 แต่เนื่องจากค่าตัววัดความมั่นคงคำนวณมาจากค่าเฉลี่ยของคะแนนซีวีเอสเอส ซึ่งเราต้องการให้มีค่าน้อย ๆ (ถ้ามีค่าน้อยแสดงว่ามีผลกระทบกับระบบน้อย) ซึ่งในที่นี้จึงมีการเบี่ยงเบนไปในแนวทางที่เราไม่ต้องการ ซึ่งก็คือ Negative นั่นเอง

4.4.3 **หน้า** Vulnerability Information

จะแสดงรายละเอียดของซอฟต์แวร์ที่ติดตั้งอยู่ในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ โดยจะแสดงเฉพาะซอฟต์แวร์ที่มีชื่อและเลขเวอร์ชันตรงกับชื่อและเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อน ซึ่งหน้า Vulnerability Information สามารถแสดงได้ดังรูปที่ 4.15

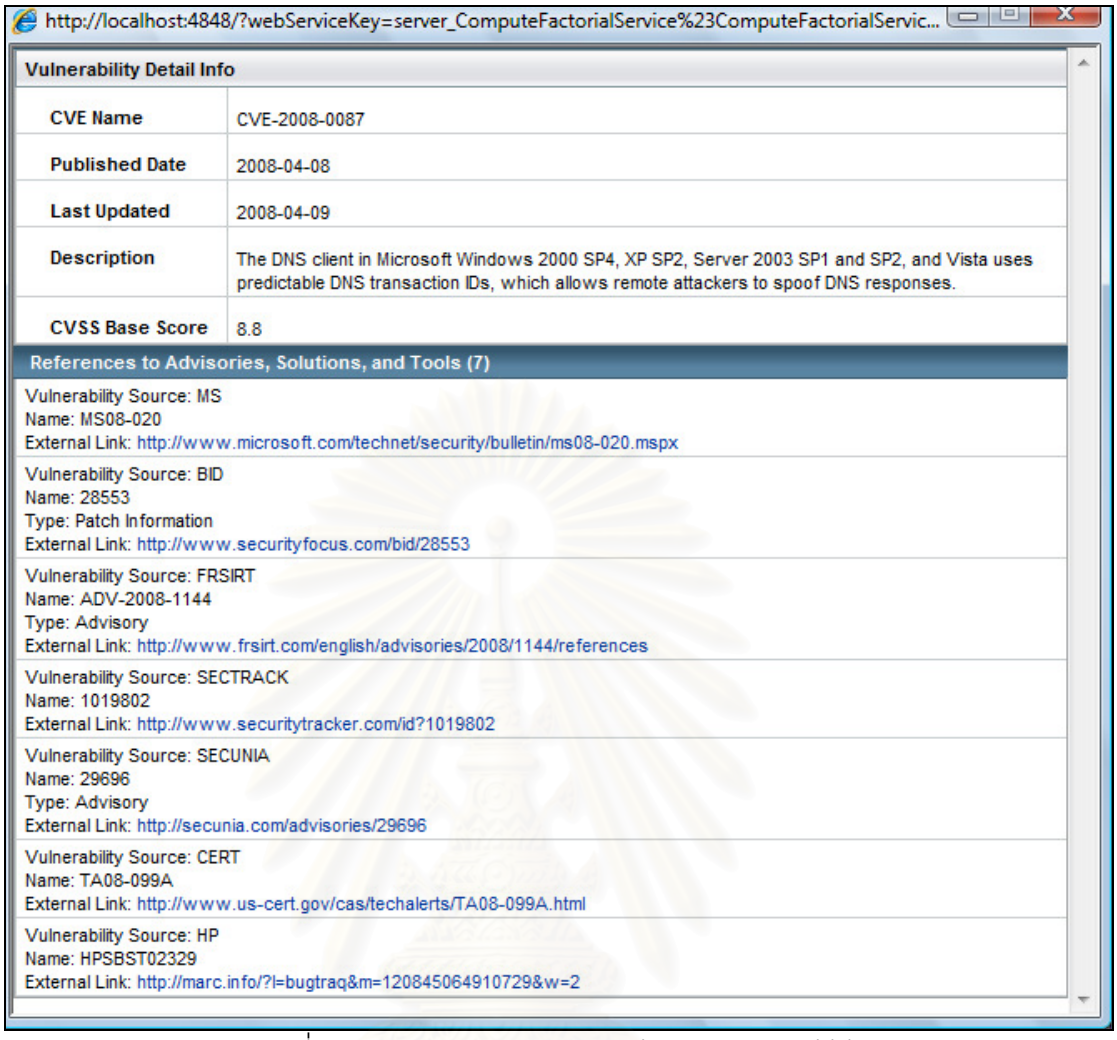


รูปที่ 4.15 หน้า Vulnerability Information

ข้อมูลที่แสดงในหน้านี้ จะอยู่ในรูปแบบตารางของข้อมูล โดยในแต่แถวจะประกอบด้วยข้อมูลดังต่อไปนี้

- ชื่อของซอฟต์แวร์
- เลขเวอร์ชัน
- รายการซีวีอีทั้งหมด ที่มีชื่อของซอฟต์แวร์และเลขเวอร์ชันที่มีจุดอ่อนในแต่ละรายการซีวีอีตรงกับชื่อและเลขเวอร์ชันของซอฟต์แวร์
- คะแนนซีวีเอสเอสเฉลี่ย ซึ่งคำนวณมาจากค่าซีวีเอสเอสของรายการซีวีอีทั้งหมด ที่มีชื่อของซอฟต์แวร์และเลขเวอร์ชันที่มีจุดอ่อนในแต่ละรายการซีวีอีตรงกับชื่อและเลขเวอร์ชันของซอฟต์แวร์

โดยรายการซีวีอีทั้งหมดที่แสดงอยู่ในตาราง ผู้ใช้สามารถคลิกเพื่อเปิดหน้าต่างที่แสดงรายละเอียดต่าง ๆ ของซีวีอีตัวนี้ ซึ่งได้แก่ ชื่อของรายการซีวีอี วันที่รายการซีวีอีนี้ถูกเผยแพร่ (Published Date) วันที่ปรับปรุงรายการซีวีอีนี้ล่าสุด (Last Updated) รายละเอียดและคะแนนซีวีเอสเอสของซีวีอีตัวนี้ นอกจากนี้ยังมีลิงค์ไปยังเว็บไซต์ที่เกี่ยวกับวิธีแก้ไขหรือคำแนะนำต่าง ๆ ที่เป็นประโยชน์เมื่อพบจุดอ่อนชนิดนี้ ดังรูปที่ 4.16



รูปที่ 4.16 หน้าต่างแสดงรายละเอียดต่าง ๆ ของซีวีอี

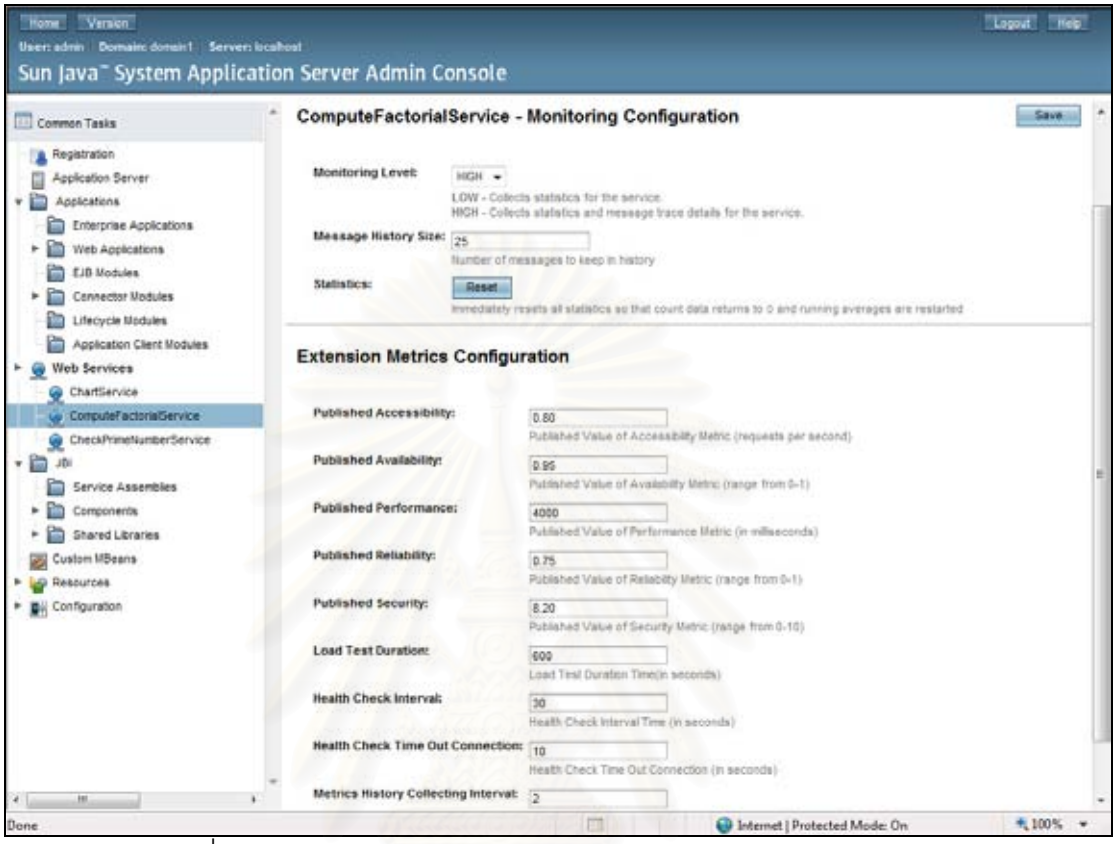
นอกจากนี้แอดมินยังสามารถปรับปรุงรายการซีวีอีที่อยู่ในส่วนจัดเก็บข้อมูลเอ็นวีดีภายในโดยใช้ข้อมูลล่าสุดจากเว็บไซต์ของเอ็นวีดี โดยคลิกที่ปุ่ม Update with Online NVD Repository ดังรูปที่ 4.17 ซึ่งจะทำการปรับปรุงรายการซีวีอีและทำการจับคู่รายการซีวีอีกับซอฟต์แวร์ที่ติดตั้งอยู่ในเครื่องที่ให้บริการเว็บเซอร์วิซ และแสดงผลลัพธ์ที่ได้ล่าสุดในหน้าจอเดิม

รูปที่ 4.17 ปุ่ม Update with Online NVD Repository

4.4.4 Extension Metrics Configuration ในหน้า Configuration

เพื่อให้สามารถปรับปรุงแก้ไขการทำงานของมอดูลในการเฝ้าสังเกตให้เหมาะสมกับการใช้งานของแอดมิน จึงได้เพิ่มโครงแบบ (Configuration) ในการจัดการที่เกี่ยวข้องกับมอดูลในการเฝ้า

สังเกตที่ได้สร้างขึ้น ซึ่งแอดมินสามารถทำการปรับปรุงแก้ไขให้เหมาะสมกับการใช้งานได้ โดยรายละเอียดของโครงแบบสามารถแสดงได้ดังรูปที่ 4.18



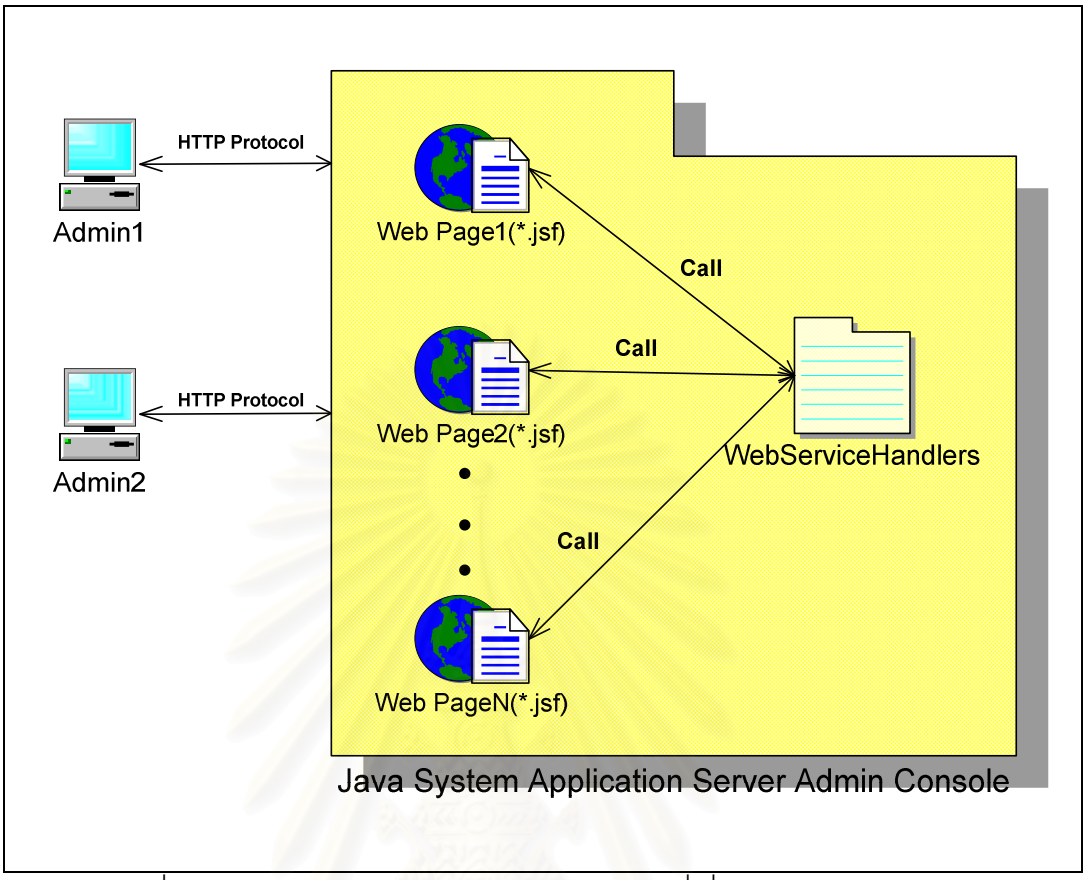
รูปที่ 4.18 Extension Metrics Configuration ในหน้า Configuration

- Published Accessibility เป็นค่าตัววัดสำหรับความสามารถในการเข้าถึงได้ ที่ได้ประกาศไว้ มีหน่วยเป็นจำนวนร้องขอต่อวินาที และจะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ
- Published Availability เป็นค่าตัววัดสำหรับสภาพพร้อมใช้งาน ที่ได้ประกาศไว้ มีค่าอยู่ระหว่าง 0-1 และจะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ
- Published Performance เป็นค่าตัววัดสำหรับสมรรถนะ ที่ได้ประกาศไว้ มีหน่วยเป็นมิลลิวินาที และถูกนำไปใช้เป็นกรอบเวลาที่คาดหวังในการคำนวณค่าตัววัดสำหรับความสามารถในการเข้าถึงได้และตัววัดสำหรับความเชื่อถือได้ นอกจากนี้ยังใช้ในการคำนวณค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบอีกด้วย

- Published Reliability เป็นค่าตัววัดสำหรับความเชื่อถือได้ ที่ได้ประกาศไว้ มีค่าอยู่ระหว่าง 0-1 และจะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ
- Published Security เป็นค่าตัววัดสำหรับความมั่นคง ที่ได้ประกาศไว้ มีค่าอยู่ระหว่าง 0-10 และจะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ
- Load Test Duration เป็นช่วงเวลาที่ใช้ในการทำการทดสอบการรับภาระงาน มีหน่วยเป็นวินาที ซึ่งจะถูกนำไปใช้ในการคำนวณค่าตัววัดสำหรับความสามารถในการเข้าถึงได้
- Health Check Interval เป็นช่วงเวลาที่มอดูลเฝ้าสังเกตจะส่งคำร้องขอไปที่เว็บเซอร์วิซเพื่อเช็คว่าเว็บเซอร์วิซยังสามารถทำงานได้อยู่หรือไม่ มีหน่วยเป็นวินาที เช่น ส่งคำร้องขอไปที่เว็บเซอร์วิซทุก ๆ 15 วินาทีเป็นต้น
- Health Check Time Out Connection เป็นระยะเวลาที่มอดูลในการเฝ้าสังเกตจะรอการตอบกลับจากเว็บเซอร์วิซ มีหน่วยเป็นวินาที ซึ่งถ้าเลยช่วงระยะเวลาที่กำหนดนี้ไปแล้วจะถือว่าเว็บเซอร์วิซมีสภาพไม่พร้อมใช้งาน
- Metrics History Collecting Interval เป็นช่วงเวลาที่มอดูลในการเฝ้าสังเกตจะเก็บค่าตัววัดต่าง ๆ ที่คำนวณได้ลงใน Metrics History Repository เพื่อนำไปวิเคราะห์หรือบริหารจัดการต่อไป มีหน่วยเป็นนาที
- Metric History Location เป็นตำแหน่งของ Metrics History Repository ในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ โดยจะถูกใช้โดยมอดูลในการเฝ้าสังเกตในการเก็บข้อมูลค่าตัววัดต่าง ๆ

4.4.5 การส่งค่าข้อมูลตัววัดต่าง ๆ ไปยังส่วนแสดงผล

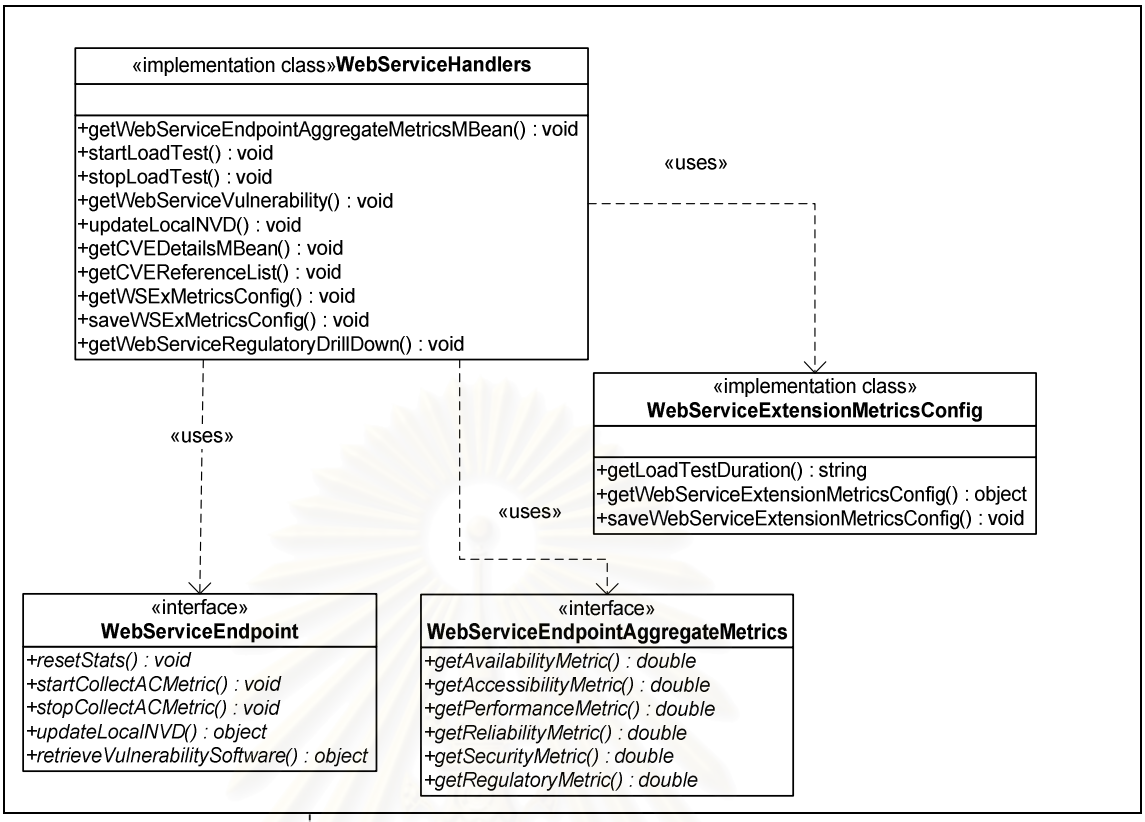
แต่ละหน้าเว็บเพจของแอดมินคอนโซลถูกสร้างขึ้นโดยใช้เทคโนโลยีจาวาเซิร์ฟเวอร์เฟซ (Java Server Faces) [23] ซึ่งช่วยอำนวยความสะดวกในการสร้างส่วนต่อประสานกับผู้ใช้ (User Interface) ในเว็บแอพพลิเคชันโดยภาษาจาวา ในจาวาเซิร์ฟเวอร์เฟซจะประกอบด้วยกลุ่มของเครื่องมือต่าง ๆ ที่ใช้ในการสร้างส่วนต่อประสานกับผู้ใช้และสามารถแทรกส่วนของคำสั่งเพื่อใช้ในการเรียกฟังก์ชันหรือมอดูลที่เขียนโดยใช้ภาษาจาวาเพื่อให้ทำงาน หรือส่งข้อมูลมายังส่วนของที่ติดต่อกับผู้ใช้ที่สร้างโดยใช้เทคโนโลยีจาวาเซิร์ฟเวอร์เฟซได้ โดยในส่วนของแอดมินคอนโซลที่เกี่ยวข้องกับเว็บเซอร์วิซซึ่งรวมถึงหน้า Extension Metrics และ Vulnerability Information มีการทำงานดังรูปที่ 4.19



รูปที่ 4.19 การทำงานของแอดมินคอนโซลในส่วนที่เกี่ยวข้องกับเว็บเซอร์วิซ

แต่ละเว็บเพจจะอยู่ในไฟล์สกุล jsf (JavaServer Faces) ซึ่งจะถูกแปลงเป็น เอชทีเอ็มแอล (HTML:Hyper Text Markup Language) ก่อนที่จะส่งไปแสดงผลที่เครื่องเป้าหมายผ่านทางเอชทีทีพีโพรโทคอล (HTTP Protocol) ฟังก์ชันการทำงานเบื้องหลังและช่องทางในการติดต่อกับมอดูลอื่น ๆ ในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์จะทำผ่านอ็อบเจกต์ WebServiceHandler ทั้งในการดึงข้อมูลมาแสดงผลและการควบคุมการทำงานของเว็บเซอร์วิซผ่านทางแอดมินคอนโซล

ในงานวิจัยนี้จะเพิ่มฟังก์ชันลงในคลาส WebServiceHandler เพื่อดึงข้อมูลค่าตัววัดต่างๆ ที่ได้จากมอดูลในการคำนวณค่าตัววัดและเพื่อควบคุมการทำงานของมอดูลในการเฝ้าสังเกต ซึ่งสามารถแสดงได้ดังรูปที่ 4.20



รูปที่ 4.20 แผนภาพคลาส WebServiceHandler

โดยค่าตัววัดที่นำมาแสดงจะดึงมาจาก WebServiceEndpointAggregateMetrics ส่วนการควบคุมการทำงานของมอดูลในการเฝ้าสังเกตจะทำผ่าน WebServiceEndpoint และการบริหารจัดการมอดูลเฝ้าสังเกตให้เหมาะกับการใช้งานจะทำผ่าน WebServiceExtensionMetricsConfig ซึ่งรายละเอียดของฟังก์ชันที่ถูกเพิ่มใน คลาส WebServiceHandler เพื่อทำงานตามหัวข้อที่ 4.4.1 – 4.4.3 สามารถแสดงได้ดังตารางที่ 4.6

ตารางที่ 4.6 ฟังก์ชันที่ถูกเพิ่มลงไปในคลาส WebServiceHandler

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
1. getWebServiceEndpoint AggregateMetricsMBean	เป็นฟังก์ชันที่ถูกใช้ในหน้า Extension Metrics ซึ่งจะให้ข้อมูลค่าตัววัดต่าง ๆ ที่ได้จากมอดูลในการคำนวณค่าตัววัด เพื่อนำมาแสดงในหน้า Extension Metrics
2. startLoadTest	เป็นฟังก์ชันที่ถูกใช้ในหน้า Extension Metrics ซึ่งจะถูกเรียกเมื่อมีการกดปุ่ม Start Load Test เพื่อบอกให้มอดูลในการเฝ้าสังเกตเริ่มเก็บข้อมูลตัววัดสำหรับความสามารถในการเข้าถึงได้

ตารางที่ 4.6 ฟังก์ชันที่ถูกเพิ่มลงไปในคลาส WebServiceHandler (ต่อ)

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
3. stopLoadTest	เป็นฟังก์ชันที่ถูกใช้ในหน้า Extension Metrics ซึ่งจะถูกเรียกเมื่อมีการกดปุ่ม Stop Load Test เพื่อบอกให้มอดูลในการเฝ้าสังเกตหยุดเก็บข้อมูล และทำการคำนวณค่าตัววัดสำหรับความสามารถในการเข้าถึงได้เพื่อนำมาแสดงในหน้า Extension Metrics
4. getWebServiceVulnerability	เป็นฟังก์ชันที่ถูกใช้ในหน้า Vulnerability Information เพื่อดึงรายการของซอฟต์แวร์ที่มีจุดอ่อน พร้อมทั้งรายการซีวีอีและคะแนนซีวีเอสเอสที่สอดคล้องกันกับแต่ละซอฟต์แวร์ เพื่อแสดงในหน้า Vulnerability Information
5. updateLocalNVD	เป็นฟังก์ชันที่ถูกใช้ในหน้า Vulnerability Information ซึ่งจะถูกเรียกเมื่อมีการกดปุ่ม Update with Online NVD Repository เพื่อปรับปรุงรายการซีวีอีในส่วนจัดเก็บข้อมูลภายในโดยใช้ข้อมูลล่าสุดจากเว็บไซต์เอ็นวีดี แล้วคำนวณค่าผลรวมของคะแนนซีวีเอสเอสของรายการซีวีอีทั้งหมดและปรับปรุงรายการของซอฟต์แวร์ที่มีจุดอ่อน พร้อมทั้งรายการซีวีอีและคะแนนซีวีเอสเอสที่สอดคล้องกันกับแต่ละซอฟต์แวร์
6. getCVEDetailsMBean	เป็นฟังก์ชันที่ถูกใช้ในหน้า Vulnerability Information เพื่อดึงรายละเอียดของซีวีอีซึ่งได้แก่ ชื่อ วันที่เผยแพร่ วันที่แก้ไขล่าสุด และรายละเอียดของจุดอ่อนที่พบ เพื่อนำไปแสดงเมื่อผู้ใช้คลิกเลือกที่รายการซีวีอีเพื่อเปิดหน้าต่างที่แสดงรายละเอียดต่าง ๆ ของซีวีอีตัวนี้
7. getCVEReferenceList	เป็นฟังก์ชันที่ถูกใช้ในหน้า Vulnerability Information เพื่อดึงรายการเชื่อมโยง (Link) ไปยังเว็บไซต์ที่เกี่ยวกับวิธีแก้ไขหรือคำแนะนำต่าง ๆ ที่เป็นประโยชน์เมื่อพบซีวีอีชนิดนี้ เพื่อนำไปแสดงเมื่อผู้ใช้คลิกเลือกที่รายการซีวีอีเพื่อเปิดหน้าต่างที่แสดงรายละเอียดต่าง ๆ ของซีวีอีตัวนี้

ตารางที่ 4.6 ฟังก์ชันที่ถูกเพิ่มลงไปในคลาส WebServiceHandler (ต่อ)

ชื่อฟังก์ชัน	รายละเอียดของฟังก์ชัน
8. getWSExMetricsConfig	เป็นฟังก์ชันที่ถูกใช้ในหน้า Configuration เพื่อดึงรายละเอียดของโครงแบบ (Configuration) ทั้งหมดที่ใช้ในการจัดการมอดูลในการเฝ้าสังเกตจากไฟล์ ws-metrics-config.xml เพื่อแสดงในหน้า Configuration ในส่วนของ Extended Metrics Configuration
9. saveWSExMetricsConfig	เป็นฟังก์ชันที่ถูกใช้ในหน้า Configuration เพื่อเก็บข้อมูลรายละเอียดของโครงแบบทั้งหมดที่ถูกแก้ไขในหน้า Configuration ในส่วนของ Extended Metrics Configuration ลงในไฟล์ ws-metrics-config.xml
10. getWebServiceRegulatory DrillDown	เป็นฟังก์ชันที่ถูกใช้ในหน้า Regulatory Drill Down เพื่อดึงรายละเอียดของค่าตัววัดต่าง ๆ ที่นำมาใช้ในการคำนวณค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 5
การทดลองและผลการทดลอง

5.1 วิธีในการทดลอง

เพื่อให้มั่นใจได้ว่ามอดูลในการเฝ้าสังเกตที่สร้างขึ้นมาสามารถทำงานได้ถูกต้องตามที่ระบุ ดังนั้นในงานวิจัยจะมีการทดสอบการทำงานของมอดูลนี้ โดยแบ่งการทดสอบออกเป็น 5 กรณีทดสอบหลัก ตามจำนวนตัววัดที่ได้พัฒนาขึ้น ยกเว้นแต่เพียงตัววัดสมรรถนะซึ่งใช้สูตรการคำนวณเดียวกับเวลาตอบสนองกลับโดยเฉลี่ย (Average Response Time) ซึ่งมีอยู่แล้วในหน้าต่างของแอดมินของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ จึงสามารถนำมาใช้ได้เลยโดยไม่ต้องคำนวณอะไรเพิ่มเติม ดังนั้นจึงไม่มีกรณีทดสอบสำหรับตัววัดสมรรถนะ

5.1.1 **สภาพแวดล้อมในการทดสอบ (Test Environment)** ในการทดสอบจะใช้เครื่องคอมพิวเตอร์ทั้งหมด 2 เครื่องดังนี้

- เครื่องแรกทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ ซึ่งติดตั้งจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ที่มีมอดูลในการเฝ้าสังเกตที่ได้พัฒนาขึ้นรวมอยู่ด้วย โดยในเครื่องนี้จะมีเว็บเซอร์วิซที่ให้บริการอยู่ 2 เว็บเซอร์วิซ ได้แก่
  - เว็บเซอร์วิซ ComputeFactorial ให้บริการในการคำนวณค่าเอ็นแฟคทอเรียล (N-Factorial) โดยรับข้อมูลขาเข้าเป็นเลขจำนวนเต็มบวกแล้วคำนวณผลลัพธ์เป็นค่าเอ็นแฟคทอเรียลกลับไป
  - เว็บเซอร์วิซ CheckPrimeNumber ให้บริการตรวจสอบว่าตัวเลขที่เข้ามาเป็นจำนวนเฉพาะหรือไม่ โดยรับข้อมูลขาเข้าเป็นเลขจำนวนเต็มบวก แล้วตอบกลับเป็นค่าตรรกกะ (True or False) ซึ่งบ่งชี้ว่าตัวเลขที่เข้ามาเป็นจำนวนเฉพาะหรือไม่

  ทั้งนี้คำร้องขอลำดับที่ 6n (เมื่อ n เป็นเลขจำนวนเต็มมีค่าตั้งแต่ 1 ขึ้นไป) ที่เข้ามาที่เว็บเซอร์วิซทั้งสองจะถูกหน่วงการทำงานเป็นเวลา 5 วินาที
- เครื่องที่สองทำหน้าที่ส่งคำร้องขอไปยังเครื่องที่หนึ่ง โดยในเครื่องนี้จะติดตั้งซอฟต์แวร์ชื่อ soapUI เวอร์ชัน 2.0.1 ที่ใช้ในการส่งคำร้องขอไปยังเว็บเซอร์วิซ ComputeFactorial และ เว็บเซอร์วิซ CheckPrimeNumber โดยตัว soapUI สามารถกำหนดจำนวนคำร้องขอและข้อมูลขาเข้าที่ส่งไปยังเว็บเซอร์วิซ รวมถึงสามารถกำหนดระยะเวลาในการส่งคำร้องขอไปยังเว็บเซอร์วิซแต่ละตัวได้

5.1.2 **สูตรการคำนวณค่าความเบี่ยงเบน** (Deviation) ในการทดสอบจะมีการเปรียบเทียบระหว่างค่าคิวโอเอสที่ควรจะเป็นกับค่าคิวโอเอสที่ได้จากการคำนวณโดยใช้มอดูลในการเฝ้าสังเกต ในรูปแบบของค่าความเบี่ยงเบนซึ่งคำนวณได้จาก

$$\textbf{ค่าความเบี่ยงเบน (D)} = \frac{\Delta Q}{Q_e} \times 100$$

โดยที่ ΔQ เป็นผลต่างระหว่างค่าคิวโอเอสที่ควรจะเป็นกับค่าคิวโอเอสที่ได้จากการคำนวณโดยใช้มอดูลในการเฝ้าสังเกต

Q_e เป็นค่าคิวโอเอสที่ควรจะเป็น

5.1.3 **การทดสอบการคำนวณตัววัดสภาพพร้อมใช้งาน** สามารถทำได้ดังนี้

1. ในหน้าต่างของแอดมิน ล้างข้อมูลการเฝ้าสังเกตที่เก็บมาในอดีตทิ้ง แล้วเริ่มทำการจับเวลา
2. ส่งคำร้องขอไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber เพื่อตรวจสอบว่ายังทำงานได้ถูกต้องหรือไม่
3. เมื่อเวลาผ่านไป 15 นาที ให้ปิดเว็บเซอร์วิซ ComputeFactorial เป็นเวลา 5 นาที ในระหว่างที่ปิดนั้นให้ส่งคำร้องขอไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber ซึ่งในขณะที่ปิดเว็บเซอร์วิซ ComputeFactorial คำร้องขอที่ส่งไปที่เว็บเซอร์วิซ ComputeFactorial ควรจะไม่ถูกประมวลผล ในขณะที่คำร้องขอที่ส่งไปที่เว็บเซอร์วิซ CheckPrimeNumber ควรได้ผลลัพธ์กลับมาเนื่องจากไม่ได้ทำการปิดเว็บเซอร์วิซ CheckPrimeNumber หลังจากครบ 5 นาทีให้เปิดเว็บเซอร์วิซ ComputeFactorial เพื่อให้ทำงานได้ต่อไป
4. ทำข้อ 3 ซ้ำอีก 2 ครั้งจนได้เวลาในการทดสอบเป็น 60 นาที
5. หลังจากนั้นตรวจสอบค่าตัววัดสภาพพร้อมใช้งานของเว็บเซอร์วิซ ComputeFactorial และ เว็บเซอร์วิซ CheckPrimeNumber ในแอดมินคอนโซล แล้วทำการบันทึกผลการทดลอง
6. คำนวณค่าความเบี่ยงเบนที่เกิดขึ้น โดยค่าสภาพพร้อมใช้งานที่ควรจะเป็นของเว็บเซอร์วิซ ComputeFactorial คือ 0.75 ส่วนของเว็บเซอร์วิซ CheckPrimeNumber คือ 1.00
7. ทำซ้ำอีก 2 ครั้ง แล้วคำนวณหาค่าความเบี่ยงเบนเฉลี่ย

5.1.4 การทดสอบการคำนวณตัววัดความสามารถในการเข้าถึงได้ สามารถทำได้ดังนี้

1. ในหน้าต่างของแอดมิน ล้างข้อมูลการเฝ้าสังเกตที่เก็บมาในอดีตทิ้ง กำหนดกรอบเวลาที่คาดหวังในแอดมินคอนโซลไว้ที่ 4 วินาทีและกำหนดช่วงเวลาในการทำการทดสอบ (Load Test) เป็นเวลา 600 วินาที หลังจากนั้นกดปุ่ม Start Load Test
2. ส่งคำร้องขอไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber อย่างต่อเนื่อง โดยส่งคำร้องขอไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber พร้อมกัน เป็นเวลา 600 วินาที
3. เมื่อคำร้องขอถูกส่งไปที่เว็บเซอร์วิซทั้งสอง ครบ 600 วินาที ให้หยุดส่งคำร้องขอไปที่เว็บเซอร์วิซทั้งสอง กดปุ่ม Stop Load Test แล้วนับจำนวนคำร้องขอทั้งหมดที่ส่งไปที่เว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber
4. คำนวณหาจำนวนคำร้องขอทั้งหมดที่ทำงานสำเร็จภายในกรอบเวลาที่คาดหวังของเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber (ซึ่งคือ 4 วินาที) การคำนวณทำได้จาก

$$K(s) = N(s) - \lfloor N(s)/6 \rfloor$$

โดยที่ K(s) เป็นจำนวนคำร้องขอทั้งหมดที่เว็บเซอร์วิซ s ทำงานสำเร็จภายในกรอบเวลาที่คาดหวัง

N(s) เป็นจำนวนคำร้องขอทั้งหมดที่ส่งไปที่เว็บเซอร์วิซ s

$\lfloor N(s)/6 \rfloor$ เป็นค่าพื้น (Floor) ของจำนวนคำร้องขอทั้งหมดหารด้วย 6

5. หลังจากนั้นตรวจสอบค่าตัววัดความสามารถในการเข้าถึงได้ของเว็บเซอร์วิซ ComputeFactorial และ เว็บเซอร์วิซ CheckPrimeNumber ในแอดมินคอนโซล แล้วทำการบันทึกผลการทดลอง
6. คำนวณค่าความเบี่ยงเบนที่เกิดขึ้น โดยค่าความสามารถในการเข้าถึงได้ที่ควรจะเป็นของเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber ควรมีค่าเท่ากับจำนวนคำร้องขอทั้งหมดที่เว็บเซอร์วิซนั้นทำงานสำเร็จภายในกรอบเวลาที่คาดหวังหารด้วย 600 วินาที
7. ทำซ้ำอีก 2 ครั้ง แล้วคำนวณหาค่าความเบี่ยงเบนเฉลี่ย

5.1.5 **การทดสอบตัววัดความเชื่อถือได้** สามารถทำการทดสอบได้ดังนี้

1. ในหน้าต่างของแอดมิน ล้างข้อมูลการเฝ้าสังเกตที่เก็บมาในอดีตทิ้ง กำหนดกรอบเวลาที่คาดหวังในแอดมินคอนโซลไว้ที่ 4 วินาที
2. ส่งคำร้องขอไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber อย่างต่อเนื่อง โดยส่งคำร้องขอไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber พร้อมกัน เป็นเวลา 600 วินาที
3. เมื่อคำร้องขอถูกส่งไปที่เว็บเซอร์วิซทั้งสอง ครบ 600 วินาที ให้หยุดส่งคำร้องขอไปที่เว็บเซอร์วิซทั้งสอง แล้วนับจำนวนคำร้องขอทั้งหมดที่ส่งไปที่เว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber
4. คำนวณหาจำนวนคำร้องขอทั้งหมดที่ทำงานสำเร็จภายในกรอบเวลาที่คาดหวังของเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber (ซึ่งคือ 4 วินาที) โดยใช้สูตรการคำนวณเดียวกันกับหัวข้อ 5.1.5
5. หลังจากนั้นตรวจสอบค่าตัววัดความเชื่อถือได้ของเว็บเซอร์วิซ ComputeFactorial และ เว็บเซอร์วิซ CheckPrimeNumber ในแอดมินคอนโซล แล้วทำการบันทึกผลการทดลอง
6. คำนวณค่าความเบี่ยงเบนที่เกิดขึ้น โดยค่าความเชื่อถือได้ที่ควรจะเป็นของเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber ควรมีค่าเท่ากับจำนวนคำร้องขอทั้งหมดที่เว็บเซอร์วิซนั้นทำงานสำเร็จภายในกรอบเวลาที่คาดหวังหารด้วยจำนวนคำร้องขอทั้งหมดที่ส่งไปที่เว็บเซอร์วิซ
7. ทำซ้ำอีก 2 ครั้ง แล้วคำนวณหาค่าความเบี่ยงเบนเฉลี่ย

5.1.6 **การทดสอบตัววัดความมั่นคง** สามารถทำการทดสอบได้ดังนี้

1. ในหน้าต่างของแอดมิน ล้างข้อมูลการเฝ้าสังเกตที่เก็บมาในอดีตทิ้ง
2. ในหน้าต่างของแอดมิน กดปุ่มปรับปรุงข้อมูล เพื่อให้มอดูลเฝ้าสังเกตทำการค้นหาซอฟต์แวร์ที่ติดตั้งในเครื่องเว็บเซิร์ฟเวอร์ และแสดงรายการของซอฟต์แวร์ที่พบจุดอ่อน รวมถึงค่าคะแนนซีวีเอสเอสโดยเฉลี่ยของแต่ละซอฟต์แวร์ที่พบจุดอ่อน
3. นำชื่อของซอฟต์แวร์และเลขเวอร์ชันของแต่ละซอฟต์แวร์ไปค้นหารายการจุดอ่อนซีวีอีที่เกี่ยวข้องจากเว็บไซต์ของเอ็นวีดีแล้วทำการเปรียบเทียบรายการซีวีอีที่ได้ กับรายการซีวีอีที่ค้นหาโดยมอดูลเฝ้าสังเกตว่าตรงกันหรือไม่ รวมถึงตรวจสอบคะแนนซีวีเอสเอสเฉลี่ยของแต่ละซอฟต์แวร์ว่าตรงกันกับค่าคะแนนซีวีเอสเอสโดยเฉลี่ยที่คำนวณโดยมอดูลเฝ้าสังเกตหรือไม่

4. นำคะแนนซีวีเอสเอสของรายการซีวีอีทั้งหมดที่ค้นพบมาคำนวณหาค่าเฉลี่ยแล้วนำค่าที่ได้ไปเปรียบเทียบกับค่าตัววัดความมั่นคงที่วัดได้จากเครื่องมือในการเฝ้าสังเกตแล้วคำนวณหาค่าความเบี่ยงเบนที่เกิดขึ้น
5. ทำซ้ำอีก 2 ครั้ง แล้วคำนวณหาค่าความเบี่ยงเบนเฉลี่ย

5.1.7 **การทดสอบตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ** สามารถทำการทดสอบได้ดังนี้

1. ในหน้าต่างของแอดมิน ล้างข้อมูลการเฝ้าสังเกตที่เก็บมาในอดีตทิ้ง กำหนดกรอบเวลาที่คาดหวังในแอดมินคอนโซลไว้ที่ 4 วินาที
2. ในหน้าต่างของแอดมิน กดปุ่มปรับปรุงข้อมูลเพื่อให้มอดูลเฝ้าสังเกตทำการค้นหาซอฟต์แวร์ที่ติดตั้งในเครื่องเว็บเซิร์ฟเวอร์ และแสดงรายการของซอฟต์แวร์ที่พบจุดอ่อน รวมถึงค่าคะแนนซีวีเอสเอสโดยเฉลี่ยของแต่ละซอฟต์แวร์ที่พบจุดอ่อน รวมถึงคำนวณค่าตัววัดความมั่นคงของเว็บเซอร์วิซด้วย
3. หลังจากนั้นกดปุ่ม Start Load Test แล้วส่งคำร้องขอไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber อย่างต่อเนื่อง โดยส่งคำร้องขอไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber พร้อมกัน เป็นเวลา 600 วินาที
4. เมื่อคำร้องขอถูกส่งไปที่เว็บเซอร์วิซทั้งสอง ครบ 600 วินาที ให้หยุดส่งคำร้องขอไปที่เว็บเซอร์วิซทั้งสอง แล้วกดปุ่ม Stop Load Test
5. ไปยังหน้าต่างของแอดมิน คำนวณหาค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบที่ควรจะเป็น โดยใช้ค่าคิวโอเอสในหน้าต่างของแอดมินเป็นข้อมูลขาเข้าของค่าคิวโอเอสที่ได้จากการเฝ้าสังเกต นำค่าที่ได้จากการคำนวณมาเปรียบเทียบกับค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบที่ได้จากการเฝ้าสังเกต แล้วคำนวณหาค่าความเบี่ยงเบน
6. ทำซ้ำอีก 2 ครั้ง แล้วคำนวณหาค่าความเบี่ยงเบนเฉลี่ย

5.2 ผลการทดลอง

จากการทดลองดังกล่าวข้างต้น ได้ผลการทดลองดังตารางที่ 5.1 ถึง ตารางที่ 5.7 ดังต่อไปนี้

ตารางที่ 5.1 ผลการทดสอบการคำนวณตัววัดสภาพพร้อมใช้งาน ที่ได้จากมอดูลในการเฝ้าสังเกตเปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.3

เว็บเซอร์วิซ		ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่าเฉลี่ย
ComputeFactorial	ค่าที่ได้จากมอดูลเฝ้าสังเกต	0.750	0.750	0.750	0.750
	ค่าที่ควรจะเป็น	0.750	0.750	0.750	0.750
	ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	0.000
CheckPrimeNumber	ค่าที่ได้จากมอดูลเฝ้าสังเกต	1.000	1.000	1.000	1.000
	ค่าที่ควรจะเป็น	1.000	1.000	1.000	1.000
	ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	0.000

ในการทดสอบการคำนวณตัววัดสภาพพร้อมใช้งาน มีการปิดการทำงานของเว็บเซอร์วิซ ComputeFactorial เป็นเวลา 10 นาทีและเปิดการทำงานของเว็บเซอร์วิซเป็นเวลา 20 นาทีสลับกันไปจนครบหนึ่งชั่วโมง ดังนั้นค่าตัววัดสภาพพร้อมใช้งานที่ควรจะเป็นคือ 0.750 ส่วนเว็บเซอร์วิซ CheckPrimeNumber ไม่มีการปิดการทำงานของเว็บเซอร์วิซเลยตลอดหนึ่งชั่วโมง ดังนั้นค่าตัววัดสภาพพร้อมใช้งานที่ควรจะเป็นคือ 1.000 ซึ่งตรงกันกับค่าที่ได้จากมอดูลในการเฝ้าสังเกตในตารางที่ 5.1 แสดงให้เห็นว่า มอดูลในการเฝ้าสังเกตที่พัฒนาขึ้นสามารถทำงานได้ถูกต้องในการคำนวณหาค่าตัววัดสภาพพร้อมใช้งานโดยไม่มีความเบี่ยงเบนเกิดขึ้น

ตารางที่ 5.2 ผลการทดสอบการคำนวณตัววัดความสามารถในการเข้าถึงได้ ที่ได้จากมอดูลในการเฝ้าสังเกตเปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.4

เว็บเซอร์วิซ		ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่าเฉลี่ย
ComputeFactorial	ค่าที่ได้จากมอดูลเฝ้าสังเกต	1.008	1.008	1.008	1.008
	ค่าที่ควรจะเป็น	1.008	1.008	1.008	1.008
	ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	0.000
CheckPrimeNumber	ค่าที่ได้จากมอดูลเฝ้าสังเกต	1.000	1.000	1.000	1.000
	ค่าที่ควรจะเป็น	1.000	1.000	1.000	1.000
	ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	0.000

ในการทดสอบการคำนวณตัววัดความสามารถในการเข้าถึงได้ จำนวนคำร้องขอที่ส่งไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber ในแต่ละครั้งที่ทดสอบภายในระยะเวลา 600 วินาที เป็นจำนวน 725 และ 720 คำร้องขอตามลำดับ ดังนั้นจำนวนคำร้องขอที่ทำงานสำเร็จภายใต้กรอบเวลา 4 วินาทีของเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber มีจำนวน 605 และ 600 คำร้องขอ ตามลำดับ จากตารางที่ 5.2 ค่าที่ควรจะเป็นที่แสดงในตารางคำนวณมาจาก จำนวนคำร้องขอที่ทำงานสำเร็จภายใต้กรอบเวลา 4 วินาทีหารด้วยระยะเวลาในการเฝ้าสังเกต ซึ่งก็คือ 600 วินาที ทำให้ได้ค่าควรจะเป็นของเว็บเซอร์วิซ

ComputeFactorial และ CheckPrimeNumber เป็น 1.008 และ 1.000 ตามลำดับ ซึ่งตรงกันกับค่าที่ได้จากมอดูลในการเฝ้าสังเกต แสดงให้เห็นว่า มอดูลในการเฝ้าสังเกตที่พัฒนาขึ้นสามารถทำงานได้ถูกต้องในการคำนวณหาค่าตัววัดความสามารถในการเข้าถึงได้โดยไม่มีความเบี่ยงเบนเกิดขึ้น

ตารางที่ 5.3 ผลการทดสอบการคำนวณตัววัดความเชื่อถือได้ ที่ได้จากมอดูลในการเฝ้าสังเกต เปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.5

เว็บเซอร์วิซ		ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่าเฉลี่ย
ComputeFactorial	ค่าที่ได้จากมอดูลเฝ้าสังเกต	0.834	0.834	0.834	0.834
	ค่าที่ควรจะเป็น	0.834	0.834	0.834	0.834
	ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	0.000
CheckPrimeNumber	ค่าที่ได้จากมอดูลเฝ้าสังเกต	0.833	0.833	0.833	0.833
	ค่าที่ควรจะเป็น	0.833	0.833	0.833	0.833
	ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	0.000

ในการทดสอบการคำนวณตัววัดความเชื่อถือได้ จำนวนคำร้องขอที่ส่งไปยังเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber ในแต่ละครั้งที่ทดสอบเป็นจำนวน 725 และ 720 คำร้องขอตามลำดับ ดังนั้นจำนวนคำร้องขอที่ทำงานสำเร็จภายใต้กรอบเวลา 4 วินาทีของเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber มีจำนวน 605 และ 600 คำร้องขอ ตามลำดับ จากตารางที่ 5.3 ค่าที่ควรจะเป็นที่แสดงในตารางคำนวณมาจาก จำนวนคำร้องขอที่ทำงานสำเร็จภายใต้กรอบเวลา 4 วินาทีหารด้วยจำนวนคำร้องขอทั้งหมด ทำให้ได้ค่าควรจะเป็นของเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber เป็น 0.834 และ 0.833 ตามลำดับ ซึ่งตรงกันกับค่าที่ได้จากมอดูลในการเฝ้าสังเกต แสดงให้เห็นว่า มอดูลในการเฝ้าสังเกตที่พัฒนาขึ้นสามารถทำงานได้ถูกต้องในการคำนวณหาตัววัดความเชื่อถือได้โดยไม่มีความเบี่ยงเบนเกิดขึ้น

ตารางที่ 5.4 ผลการทดสอบการคำนวณตัววัดความมั่นคง ที่ได้จากมอดูลในการเฝ้าสังเกต เปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.6

เว็บเซอร์วิซ		ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่าเฉลี่ย
ComputeFactorial	ค่าที่ได้จากมอดูลเฝ้าสังเกต	7.447	7.447	7.447	7.447
	ค่าที่ควรจะเป็น	7.635	7.635	7.635	7.635
	ค่าความเบี่ยงเบน (%)	2.462	2.462	2.462	2.462
CheckPrimeNumber	ค่าที่ได้จากมอดูลเฝ้าสังเกต	7.447	7.447	7.447	7.447
	ค่าที่ควรจะเป็น	7.635	7.635	7.635	7.635
	ค่าความเบี่ยงเบน (%)	2.462	2.462	2.462	2.462

ในการทดสอบการคำนวณตัววัดความมั่นคง ค่าที่ควรจะเป็นที่แสดงในตารางที่ 5.4 คำนวณมาจากค่าเฉลี่ยของคะแนนซีวีเอสเอสของซีวีอีทุกตัวในเว็บไซต์เอ็นวีดีที่มีชื่อของซอฟต์แวร์และเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อน ตรงกันกับ ชื่อของซอฟต์แวร์และเลขเวอร์ชันของซอฟต์แวร์ในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซ ซึ่งจากการคำนวณได้ค่าเฉลี่ยเป็น 7.635 โดยเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber จะมีค่าเฉลี่ยของคะแนนซีวีเอสเอสเท่ากันเนื่องจากเว็บเซอร์วิซทั้งสองถูกติดตั้งลงในเครื่องเซิร์ฟเวอร์เครื่องเดียวกัน ซึ่งเมื่อเปรียบเทียบกับค่าที่ได้จากมอดูลในการเฝ้าสังเกตจะเห็นว่ามีความเบี่ยงเบนอยู่ที่ 2.462% ซึ่งจะกล่าวถึงสาเหตุของความเบี่ยงเบนที่เกิดขึ้นในหัวข้อที่ 5.3 ต่อไป

ตารางที่ 5.5 ผลการทดสอบการคำนวณตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ ที่ได้จากมอดูลในการเฝ้าสังเกตเปรียบเทียบกับค่าที่ควรจะเป็นโดยใช้กรณีทดสอบในหัวข้อที่ 5.1.7

เว็บเซอร์วิซ		**ครั้งที่** 1	**ครั้งที่** 2	**ครั้งที่** 3	**ค่าเฉลี่ย**
ComputeFactorial	ค่าที่ได้จากมอดูลเฝ้าสังเกต	0.405	0.405	0.405	0.405
	ค่าที่ควรจะเป็น	0.405	0.405	0.405	0.405
	ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	0.000
CheckPrimeNumber	ค่าที่ได้จากมอดูลเฝ้าสังเกต	0.351	0.351	0.351	0.351
	ค่าที่ควรจะเป็น	0.351	0.351	0.351	0.351
	ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	0.000

ค่าที่ได้จากตารางที่ 5.5 ทั้งค่าที่ได้คำนวณจากมอดูลในการเฝ้าสังเกตและค่าที่ควรจะเป็นเป็นค่าที่คำนวณมาจากตัววัดอื่น ๆ ในหัวข้อที่ 3.1.1 โดยใช้สูตรในการคำนวณในหัวข้อที่ 3.1.6 ซึ่งค่าของตัววัดที่ผู้ให้บริการเว็บเซอร์วิซได้ประกาศไว้ จะนำมาจากค่าตัววัดที่ได้ประกาศไว้ในแอดมินคอนโซลของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์เพื่อนำมาใช้ในการหาค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบต่อไป ตารางที่ 5.6 และ 5.7 จะแสดงรายละเอียดของค่าตัววัดต่าง ๆ ที่กำหนดไว้ในแอดมินคอนโซล ของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์และค่าที่ได้จากการเฝ้าสังเกตเพื่อแสดงให้เห็นว่าค่าที่ได้ในตารางที่ 5.5 คำนวณมาได้อย่างไร ซึ่งจากตารางที่ 5.5 จะเห็นว่าค่าที่ได้จากมอดูลในการเฝ้าสังเกตและค่าที่ควรจะเป็นมีค่าเท่ากัน แสดงให้เห็นว่ามอดูลในการเฝ้าสังเกตที่พัฒนาขึ้นสามารถทำงานได้ถูกต้องในการคำนวณหาตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบโดยไม่มีความเบี่ยงเบนเกิดขึ้น

ตารางที่ 5.6 ค่าของตัววัดแต่ละชนิดของเว็บเซอร์วิซ ComputeFactorial ที่ได้กำหนดไว้ในแอดมินคอนโซล และค่าของตัววัดแต่ละชนิดที่วัดได้โดยใช้มอดูลในการเฝ้าสังเกต

	Availability	Accessibility	Performance	Reliability	Security
ค่าที่ประกาศไว้ในแอดมินคอนโซล	0.950	0.800	4000	0.750	8.200
ค่าที่ได้จากการเฝ้าสังเกตครั้งที่ 1	1.000	1.008	830	0.834	7.447
ค่าที่ได้จากการเฝ้าสังเกตครั้งที่ 2	1.000	1.008	831	0.834	7.447
ค่าที่ได้จากการเฝ้าสังเกตครั้งที่ 3	1.000	1.008	830	0.834	7.447

ตารางที่ 5.7 ค่าของตัววัดแต่ละชนิดของเว็บเซอร์วิซ CheckPrimeNumber ที่ได้กำหนดไว้ในแอดมินคอนโซล และค่าของตัววัดแต่ละชนิดที่วัดได้โดยใช้มอดูลในการเฝ้าสังเกต

	Availability	Accessibility	Performance	Reliability	Security
ค่าที่ประกาศไว้ในแอดมินคอนโซล	0.880	0.850	4000	0.800	9.200
ค่าที่ได้จากการเฝ้าสังเกตครั้งที่ 1	1.000	1.000	843	0.833	7.447
ค่าที่ได้จากการเฝ้าสังเกตครั้งที่ 2	1.000	1.000	843	0.833	7.447
ค่าที่ได้จากการเฝ้าสังเกตครั้งที่ 3	1.000	1.000	842	0.833	7.447

5.3 วิเคราะห์ผลการทดลอง

จากผลการทดลองในหัวข้อที่ 5.2 สามารถนำมาสร้างตารางสรุปผลการทดสอบความถูกต้องในการทำงานของมอดูลในการเฝ้าสังเกตที่ได้พัฒนาขึ้น โดยทำการเปรียบเทียบค่าความเบี่ยงเบนที่เกิดขึ้นในการคำนวณค่าตัววัดแต่ละชนิดจาก 5 กรณีทดสอบของเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber ดังตารางที่ 5.8 และตารางที่ 5.9 ตามลำดับ

ตารางที่ 5.8 สรุปผลการทดสอบความถูกต้องในการทำงานของมอดูลในการเฝ้าสังเกตที่ได้พัฒนาขึ้นในแต่ละกรณีทดสอบของเว็บเซอร์วิซ ComputeFactorial

	Availability	Accessibility	Reliability	Security	Regulatory
ค่าที่ได้จากมอดูลเฝ้าสังเกต	0.750	1.008	0.834	7.447	0.405
ค่าที่ควรจะเป็น	0.750	1.008	0.834	7.635	0.405
ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	2.462	0.000

ตารางที่ 5.9 สรุปผลการทดสอบความถูกต้องในการทำงานของมอดูลในการเฝ้าสังเกตที่ได้พัฒนาขึ้นในแต่ละกรณีทดสอบของเว็บเซอร์วิซ CheckPrimeNumber

	Availability	Accessibility	Reliability	Security	Regulatory
ค่าที่ได้จากมอดูลเฝ้าสังเกต	1.000	1.000	0.833	7.447	0.351
ค่าที่ควรจะเป็น	1.000	1.000	0.833	7.635	0.351
ค่าความเบี่ยงเบน (%)	0.000	0.000	0.000	2.462	0.000

จากตารางที่ 5.8 และ 5.9 แสดงให้เห็นว่า เครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้นสามารถทำงานได้ถูกต้องในการคำนวณค่าตัววัดสภาพพร้อมใช้งาน ค่าตัววัดความสามารถในการเข้าถึงได้ ค่าตัววัดความเชื่อถือได้ และค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบโดยที่ไม่มีความเบี่ยงเบนเกิดขึ้น นอกจากนี้เครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้นยังรองรับการเฝ้าสังเกตในหลาย ๆ เว็บเซอร์วิซพร้อม ๆ กันและสามารถทำงานได้อย่างถูกต้อง ซึ่งจะเห็นได้จากผลการทดลองที่สอดคล้องกันในการทดสอบเปรียบเทียบการเฝ้าสังเกตของเว็บเซอร์วิซ ComputeFactorial และ CheckPrimeNumber ซึ่งให้ค่าความเบี่ยงเบนที่ตรงกันคือเป็น 0% ในทุกๆตัววัด ยกเว้นค่าตัววัดความมั่นคงที่ให้ค่าความเบี่ยงเบนเท่ากันคือ 2.462%

ในตารางที่ 5.5 ค่าตัววัดความมั่นคงซึ่งคำนวณมาจากเครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้นนั้น มีความเบี่ยงเบนเกิดขึ้นเมื่อเทียบกับค่าที่ควรจะเป็น ซึ่งเมื่อทำการวิเคราะห์ไปในรายละเอียดตามรายการของซอฟต์แวร์แต่ละตัวที่พบจุดอ่อน พบว่าการจับคู่ระหว่างซอฟต์แวร์ที่ติดตั้งในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซกับรายการซีวีอีนั้นมีความถูกต้องโดยรวมประมาณ 96.86% ดังตารางที่ 5.10

ตารางที่ 5.10 ผลลัพธ์ในการจับคู่รายการซีวีอีในเครื่องเป้าหมายโดยใช้มอดูลในการเฝ้าสังเกตเทียบกับรายการซีวีอีที่ค้นจากเว็บไซต์ของเอ็นวีดี

ชื่อของซอฟต์แวร์	**เลขเวอร์ชัน**	**ผลการจับคู่**		**รวม**	**%ความถูกต้อง**
		ถูกต้อง	**ไม่ถูกต้อง**		
1. Windows Vista	vista	60	0	60	100
2. Internet Explorer 7.0	7.0	92	1	93	98.92
3. Adobe Flash Player Plugin	9.0.124.0	2	1	3	66.67
4. GDR 3068 for SQL Server Database Services 2005 ENU	9.2.3068	10	0	10	100
5. GDR 3068 for SQL Server Tools and Workstation Components 2005 ENU	9.2.3068	10	0	10	100
6. Microsoft SQL Server 2005	--	10	0	10	100

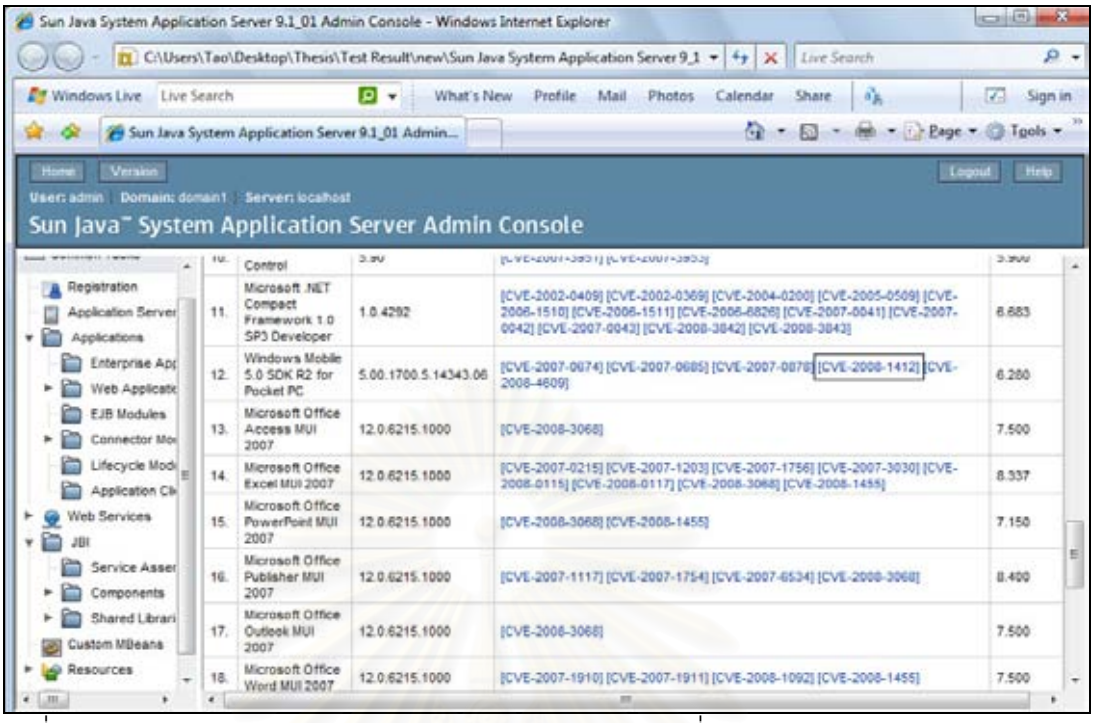
ตารางที่ 5.10 ผลลัพธ์ในการจับคู่รายการซีวีอีในเครื่องเป้าหมายโดยใช้มอดูลในการเฝ้าสังเกตเทียบกับรายการซีวีอีที่ค้นจากเว็บไซต์ของเอ็นวีดี (ต่อ)

ชื่อของซอฟต์แวร์	เลขเวอร์ชัน	ผลการจับคู่		รวม	%ความถูกต้อง
		ถูกต้อง	ไม่ถูกต้อง		
7. Microsoft SQL Server 2005 Mobile [ENU] Developer Tools	3.0.0.0	10	0	10	100
8. Microsoft SQL Server 2005 Tools Express Edition	9.2.3042.00	10	0	10	100
9. Microsoft SQL Server 2005 Express Edition	9.2.3042.00	10	0	10	100
10. Norman Virus Control	5.90	2	0	2	100
11. Microsoft .NET Compact Framework 1.0 SP3 Developer	1.0.4292	11	1	12	91.67
12. Windows Mobile 5.0 SDK R2 for Pocket PC	5.00.1700. 5.14343.06	4	1	5	80
13. Microsoft Office Access MUI 2007	12.0.6215.1000	1	0	1	100
14. Microsoft Office Excel MUI 2007	12.0.6215.1000	8	0	8	100
15. Microsoft Office PowerPoint MUI 2007	12.0.6215.1000	2	0	2	100
16. Microsoft Office Publisher MUI 2007	12.0.6215.1000	4	0	4	100
17. Microsoft Office Outlook MUI 2007	12.0.6215.1000	1	0	1	100
18. Microsoft Office Word MUI 2007	12.0.6215.1000	4	0	4	100
19. Microsoft Office InfoPath MUI 2007	12.0.6215.1000	1	0	1	100
20. Microsoft Office OneNote MUI 2007	12.0.6215.1000	1	0	1	100
21. Microsoft Office Groove MUI 2007	12.0.6215.1000	1	0	1	100
22. Microsoft Office Groove Setup Metadata MUI 2007	12.0.6215.1000	1	0	1	100
23. Microsoft Office Visio Professional 2003	11.0.8173.0	7	4	11	63.64
24. Windows Mobile 5.0 SDK R2 for Smartphone	5.00.1700. 5.14343.06	4	1	5	80
25. Adobe Reader 8.1.2	8.1.2	3	0	3	100
26. DivX Converter	6.6.0	1	0	1	100
27. Microsoft .NET Compact Framework 2.0 SP2	2.0.7045	8	0	8	100
รวม		**278**	**9**	**287**	**96.86**

จากตารางที่ 5.10 มีรายการซีวีอีทั้งหมด 9 รายการที่เป็นการจับคู่ที่ไม่ถูกต้องหรือคิดเป็น 3.14% ของรายการซีวีอีทั้งหมดที่เป็นผลลัพธ์ในการจับคู่ของมอดูลในการเฝ้าสังเกตที่ได้พัฒนาขึ้น ซึ่งส่งผลให้การคำนวณค่าตัววัดความมั่นคงของมอดูลในการเฝ้าสังเกตเบี่ยงเบน ซึ่งการจับคู่ที่ไม่ถูกต้องนั้นมีอยู่ด้วยกัน 2 กรณี ดังนี้

1) รายการซีวีอีที่นำมาใช้คำนวณค่าตัววัดความมั่นคงไม่ใช่รายการซีวีอีที่ถูกต้อง กล่าวคือเป็นรายการซีวีอีที่ไม่ได้มีชื่อและเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อน ตรงกันกับ ชื่อและเลขเวอร์ชันของซอฟต์แวร์ที่ติดตั้งในเครื่องที่ให้บริการเว็บเซอร์วิซ แต่เครื่องมือในการเฝ้าสังเกตกลับนำเอารายการซีวีอีตัวนี้มาใช้ในการคำนวณค่าตัววัดความมั่นคง จึงความผิดแบบ False Positive ซึ่งมีสาเหตุมาจากขั้นตอนวิธี (Algorithm) ในการจับคู่ซอฟต์แวร์กับรายการซีวีอี ที่ใช้วิธีแบ่งชื่อซอฟต์แวร์ออกเป็นคำ ๆ แล้วนำมาจับคู่ ซึ่งชื่อซอฟต์แวร์ที่มีจุดอ่อนบางชื่อประกอบด้วยคำหลาย ๆ คำ ทำให้อาจเกิดกรณีที่คำที่ตัดมาเพื่อทำการจับคู่ตรงกับชื่อของซอฟต์แวร์ที่มีจุดอ่อน และ ยังมีเลขเวอร์ชันตรงกัน ทำให้มอดูลในการเฝ้าสังเกตบันทึกรายการซีวีอีตัวนี้ลงในรายการซีวีอีที่เป็นผลกระทบจากการติดตั้งซอฟต์แวร์ที่นำมาจับคู่ ดังตัวอย่างในรูปที่ 5.1 และ รูปที่ 5.2

รูปที่ 5.1 รายละเอียดของรายการซีวีอี CVE-2008-1412

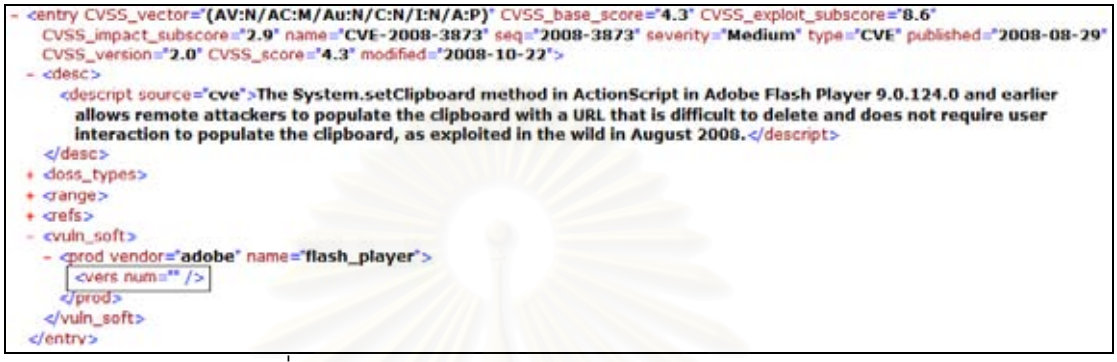


รูปที่ 5.2 หน้า Vulnerability Information ในแอดมินคอนโซลที่แสดงรายการซอฟต์แวร์ Windows Mobile 5.0 SDK R2 for Pocket PC

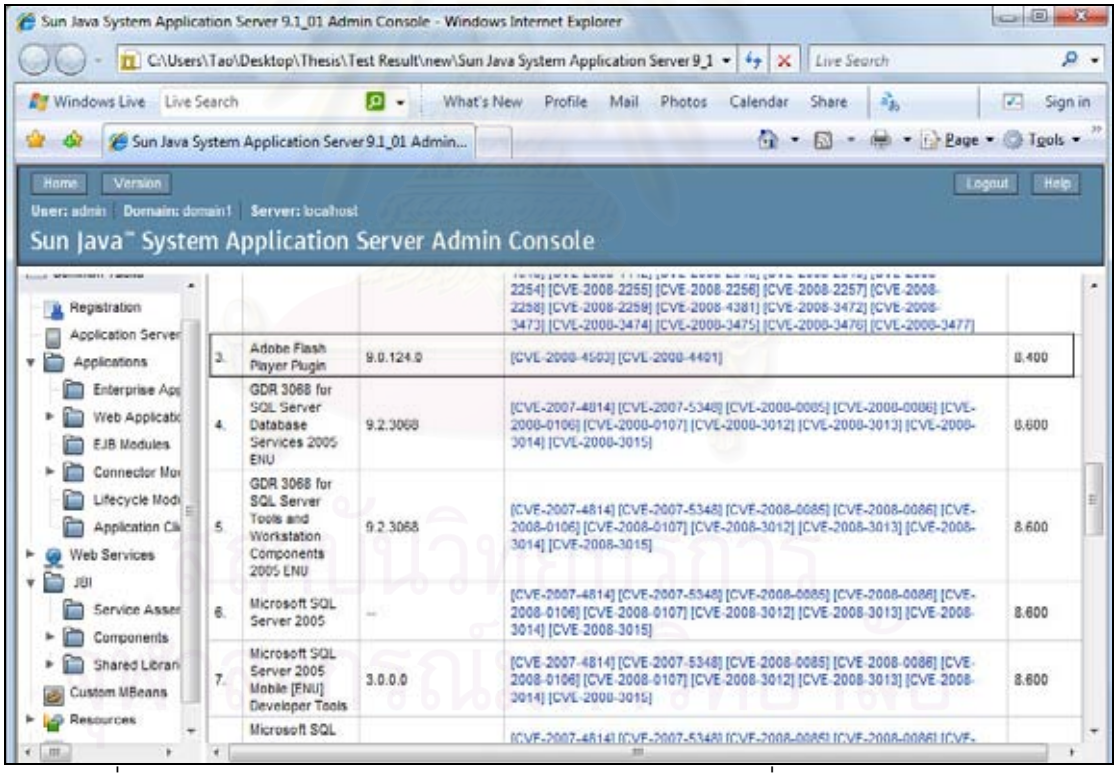
ในรูปที่ 5.1 เป็นรายละเอียดของรายการซีวีอี CVE-2008-1412 ส่วนรูปที่ 5.2 เป็นหน้า Vulnerability Information ในแอดมินคอนโซล ซึ่งจะแสดงรายละเอียดของซอฟต์แวร์ที่พบจุดอ่อน จากรูปที่ 5.2 จะเห็นว่า ซอฟต์แวร์ Windows Mobile 5.0 SDK R2 for Pocket PC ในลำดับที่ 12 มีการระบุว่ารายการซีวีอี CVE-2008-1412 เป็นผลกระทบของการติดตั้ง ซอฟต์แวร์ Windows Mobile 5.0 SDK R2 for Pocket PC แต่ในความเป็นจริงนั้นเมื่อพิจารณาในรูปที่ 5.1 ไม่พบว่าซอฟต์แวร์ Windows Mobile 5.0 SDK R2 for Pocket PC เป็นชื่อของซอฟต์แวร์ที่พบจุดอ่อน CVE-2008-1412 ทั้งนี้เนื่องจากชื่อของซอฟต์แวร์ Windows Mobile 5.0 SDK R2 for Pocket PC เมื่อถูกตัดเป็นคำ ๆ แล้วนำไปจับคู่โดยใช้มอดูลในการเฝ้าสังเกต คำว่า mobile จะตรงกับชื่อซอฟต์แวร์ f-secure_mobile_antivirus_for_windows_mobile ซึ่งเป็นชื่อของซอฟต์แวร์ที่มีจุดอ่อน CVE-2008-1412 และเลข 5.0 ยังตรงกับเลขเวอร์ชันของซอฟต์แวร์ f-secure_mobile_antivirus_for_windows_mobile ทำให้มอดูลในการเฝ้าสังเกตบันทึกรายการซีวีอี CVE-2008-1412 ว่าเป็นผลกระทบของการติดตั้ง ซอฟต์แวร์ Windows Mobile 5.0 SDK R2 for Pocket PC

2) รายการซีวีอีที่เป็นผลกระทบของซอฟต์แวร์ที่ติดตั้งในเครื่องให้บริการเว็บเซอร์วิซบางรายการ ไม่ถูกนำมาใช้ในการคำนวณค่าตัววัดความมั่นคง ทั้งๆที่ควรนำมาใช้ในการ

คำนวณ จึงความผิดแบบ False Negative ซึ่งมีสาเหตุมาจากข้อมูลรายละเอียดรายการซีวีอีในเว็บไซต์ของเอ็นวีดีไม่สมบูรณ์ บางรายการซีวีอีระบุเฉพาะชื่อของซอฟต์แวร์ที่มีจุดอ่อน แต่ไม่ระบุเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อน ทำให้การค้นหาของมอดูลวิเคราะห์ความมั่นคงไม่สามารถจับคู่กับรายการซีวีอีในกรณีนี้ได้ ดังตัวอย่างใน รูปที่ 5.3 และ รูปที่ 5.4



รูปที่ 5.3 รายละเอียดของรายการซีวีอี CVE-2008-3873



รูปที่ 5.4 หน้า Vulnerability Information ในแอดมินคอนโซลที่แสดงรายการซอฟต์แวร์ Adobe Flash Player Plugin

ในรูปที่ 5.3 เป็นรายละเอียดของรายการซีวีอี CVE-2008-3873 ซึ่งถ้าอ่านในรายละเอียดของรายการซีวีอีนี้ จะพบว่ารายการซีวีอีนี้เป็นผลกระทบของการติดตั้ง

ซอฟต์แวร์ Adobe Flash Player Plugin แต่เมื่อพิจารณาในรูปที่ 5.4 ซึ่งเป็นหน้า Vulnerability Information ในแอดมินคอนโซลที่แสดงรายการซอฟต์แวร์ Adobe Flash Player Plugin ในลำดับที่ 3 จะพบว่ามอดูลในการเฝ้าสังเกตไม่บันทึกรายการซีวีอี CVE-2008-3873 ว่าเป็นผลกระทบของการติดตั้งซอฟต์แวร์ เพราะรายการซีวีอีนี้ไม่มีรายละเอียดของเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อน จึงไม่ได้ถูกนำมาบันทึก

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะ

งานวิจัยนี้เป็นการศึกษาเกี่ยวกับการเฝ้าสังเกตคุณภาพในการให้บริการของเว็บเซอร์วิซ โดยเน้นไปที่การเฝ้าสังเกตในฝั่งของผู้ให้บริการเว็บเซอร์วิซ เพื่อนำไปใช้ในการปรับปรุงคุณภาพในการให้บริการให้ดีขึ้น โดยมีการกำหนดตัววัดในแง่มุมต่าง ๆ และกำหนดข้อมูลในการเฝ้าสังเกตเพื่อให้ได้มาซึ่งค่าตัววัดเหล่านั้น นอกจากนี้ยังได้พัฒนาเครื่องมือในการเฝ้าสังเกตและคำนวณค่าตัววัดต่าง ๆ ที่ได้กำหนดขึ้นโดยทำการขยายจากจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์

6.1 สรุปผลการวิจัย

จากการศึกษา ออกแบบ และพัฒนาในส่วนของตัววัดคุณภาพในการให้บริการเว็บเซอร์วิซ และในส่วนของเครื่องมือในการเฝ้าสังเกตและคำนวณค่าตัววัด รวมไปถึงการทดสอบการทำงานของเครื่องมือในการเฝ้าสังเกตและคำนวณค่าตัววัดที่ได้สร้างขึ้น สามารถสรุปผลการวิจัยได้ดังต่อไปนี้

6.1.1 เครื่องมือในการเฝ้าสังเกตและคำนวณค่าตัววัดที่ได้พัฒนาขึ้นมา สามารถทำงานได้ถูกต้องแม่นยำในระดับที่น่าพอใจ จากกรณีทดสอบทั้ง 5 กรณีค่าตัววัดที่คำนวณโดยใช้เครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้นไม่มีความเบี่ยงเบนเกิดขึ้น ยกเว้นค่าตัววัดสำหรับความมั่นคงที่พบความเบี่ยงเบนเล็กน้อย จากสาเหตุที่กล่าวในหัวข้อที่ 5.3

6.1.2 การจับคู่ซอฟต์แวร์ที่ติดตั้งในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซกับรายการซีวีอี โดยการแบ่งชื่อของซอฟต์แวร์ออกเป็นคำ ๆ ช่วยเพิ่มโอกาสในการค้นหาหรือการจับคู่กับรายการซีวีอีให้มากขึ้น แต่อาจทำให้เกิดข้อผิดพลาดหรือเบี่ยงเบนในกรณีที่ชื่อของซอฟต์แวร์ที่มีจุดอ่อนอาจมีส่วนหนึ่งของชื่อตรงกับคำที่นำมาค้นหา ซึ่งในงานวิจัยนี้พยายามแก้ไขปัญหานี้โดยการตรวจสอบเลขเวอร์ชันของซอฟต์แวร์ที่มีจุดอ่อนกับเลขเวอร์ชันของซอฟต์แวร์ที่มีส่วนของชื่อตรงกันซึ่งจะเป็นการกรองเอารายการซีวีอีที่มีส่วนหนึ่งของชื่อตรงกับคำที่ค้นหาออกไปส่วนหนึ่ง นอกจากนี้ยังมีการกรองเอาเฉพาะคำที่มีนัยสำคัญในการค้นหามาใช้ในการจับคู่เท่านั้น เพื่อให้การจับคู่มีความแม่นยำสูงสุด

6.1.3 ค่าตัววัดที่ได้จากการคำนวณโดยใช้เครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้น สามารถช่วยผู้ให้บริการเว็บเซอร์วิซได้ทราบและเข้าใจถึงการทำงานของเว็บเซอร์วิซของตนในแง่มุมต่าง ๆ ได้แก่

- ทราบว่าเว็บเซอร์วิซของตนมีความเสถียรหรือพร้อมใช้งานอยู่ตลอดเวลาหรือไม่จากค่าตัววัดสำหรับสภาพพร้อมใช้งาน

- ทราบว่าจำนวนคำร้องขอที่เว็บเซอร์วิซของตนสามารถทำงานได้ถูกต้องหรือภายใต้กรอบเวลาที่คาดหวังภายในช่วงเวลาหนึ่งมีค่าเป็นเท่าไรจากตัววัดสำหรับความสามารถในการเข้าถึงได้
- ทราบว่าเวลาประมวลผลโดยเฉลี่ยเป็นเท่าไรจากค่าตัววัดสมรรถนะ
- ทราบว่าความน่าจะเป็นโดยเฉลี่ยที่เว็บเซอร์วิซของตนสามารถทำงานได้ถูกต้องหรือภายใต้กรอบเวลาคาดหวังเป็นเท่าไรเมื่อเทียบกับจำนวนคำร้องขอที่เข้ามาทั้งหมดจากตัววัดสำหรับความเชื่อถือได้
- ทราบว่าความรุนแรงของจุดอ่อนโดยเฉลี่ยที่พบในเครื่องเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซของตนมีค่าเป็นเท่าไร
- ทราบว่าการทำงานของเว็บเซอร์วิซของตนโดยรวมเป็นไปตามที่ได้ประกาศหรือกำหนดไว้หรือไม่จากค่าตัววัดสำหรับการควบคุมดูแลให้เป็นไปตามกฎระเบียบ

6.1.4 การออกแบบและพัฒนาเครื่องมือในการเฝ้าสังเกตและคำนวณค่าตัววัด โดยทำการขยายจากมอดูลในการเฝ้าสังเกตของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ ทำให้เว็บเซอร์วิซที่ติดตั้งอยู่ในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์สามารถใช้งานเครื่องมือในการเฝ้าสังเกตและคำนวณค่าตัววัดที่ได้พัฒนาขึ้น โดยไม่ต้องติดตั้งซอฟต์แวร์ตัวอื่นลงไปด้วย และยังเป็นการเพิ่มความสามารถในการเฝ้าสังเกตของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ให้มากกว่าและครอบคลุมแง่มุมต่าง ๆ ของคุณภาพในการให้บริการกว่าเครื่องมือในการเฝ้าสังเกตที่มีอยู่ในปัจจุบัน

6.2 ปัญหาและข้อจำกัดของงานวิจัย

6.2.1 เครื่องมือในการเฝ้าสังเกตและคำนวณค่าตัววัดที่ได้พัฒนาขึ้นมา สามารถใช้งานได้ในระบบปฏิบัติการวินโดวส์เท่านั้น เนื่องจากใช้การดึงชื่อของซอฟต์แวร์ที่ติดตั้งอยู่ในเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซจากวินโดวส์รีจิสทรี

6.2.2 เครื่องมือในการเฝ้าสังเกตและคำนวณค่าตัววัดที่ได้พัฒนาขึ้นมา ไม่สามารถเฝ้าสังเกตหรือเก็บข้อมูลบางประการที่ทำได้เฉพาะในฝั่งของผู้ใช้บริการเว็บเซอร์วิซเท่านั้น เช่น เวลาในการตอบสนองกลับ (Response Time) เป็นต้น เนื่องจากถูกพัฒนามาเพื่อเฝ้าสังเกตคุณภาพในการให้บริการในฝั่งเซิร์ฟเวอร์เท่านั้น จึงควรจะมีการเสริมด้วยการเฝ้าสังเกตคุณภาพในการให้บริการในฝั่งผู้ใช้บริการเว็บเซอร์วิซด้วย เพื่อให้ครอบคลุมแง่มุมต่าง ๆ ในการเฝ้าสังเกตคุณภาพในการให้บริการได้มากขึ้น

6.2.3 ในการปรับปรุงข้อมูลรายการซีวีอีโดยใช้ข้อมูลล่าสุดจากเว็บไซต์เอ็นวีดี เวลาที่ใช้ในการปรับปรุงจะขึ้นอยู่กับแบนด์วิดท์ (Bandwidth) ของระบบเครือข่าย (Network) ที่เครื่องเซิร์ฟเวอร์นั้นใช้อยู่ ถ้าแบนด์วิดท์ของระบบเครือข่ายที่ใช้อยู่ต่ำอาจทำให้เวลาที่ใช้ในการปรับปรุงข้อมูลค่อนข้างมาก

6.2.4 การจับคู่ซอฟต์แวร์ที่ติดตั้งอยู่ในเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซกับรายการซีวีอี โดยการแบ่งชื่อซอฟต์แวร์เป็นคำ ๆ แล้วนำมาค้นหา อาจเกิดความผิดพลาดได้ถ้าชื่อของซอฟต์แวร์ในรายการซีวีอีนั้น มีส่วนของคำที่นำไปค้นหาปรากฏอยู่ด้วย นอกจากนี้อาจใช้เวลาในการทำงานค่อนข้างมากถ้าหน่วยประมวลผลของเครื่องเซิร์ฟเวอร์มีความเร็วต่ำเพราะรายการซีวีอีที่นำมาจับคู่มีจำนวนมาก จึงควรใช้กับเครื่องที่มีประสิทธิภาพสูง เช่นเครื่องระดับเซิร์ฟเวอร์ (Server Class) เพื่อไม่ให้คุณภาพในการให้บริการหรือประสิทธิภาพในการทำงานลดลงเมื่อทำการเฝ้าสังเกต

6.2.5 การที่ใช้ตัววัดเพียงหนึ่งตัวในการแทนแต่ละแง่มุมในการให้บริการเว็บเซอร์วิซอาจจะไม่เพียงพอ เช่นในงานวิจัยใช้เวลาในการประมวลผลเป็นตัวแทนของตัววัดสำหรับสมรรถนะแต่ผู้ให้บริการเว็บเซอร์วิซอาจต้องการพิจารณาปริมาณงาน (Throughput) ควบคู่ไปด้วย เพราะถ้าได้ปริมาณงานมากแต่ใช้เวลาประมวลผลน้อยก็จะทำให้เพิ่มความเชื่อมั่นว่าเว็บเซอร์วิซของตนมีสมรรถนะที่ดี

6.3 **ข้อเสนอแนะ**

แนวทางในการวิจัยหรือพัฒนาต่อจากงานวิจัยนี้ในอนาคต สามารถทำได้ 5 แนวทางดังนี้

6.3.1 พัฒนาการทำงานของเครื่องมือที่ใช้ในการเฝ้าสังเกตและคำนวณค่าตัววัดให้รองรับได้หลาย ๆ ระบบปฏิบัติการ เพื่อให้สามารถนำไปใช้ได้ในวงกว้างมากขึ้น

6.3.2 เพิ่มตัววัดในแง่มุมอื่น ๆ เช่น ในแง่ของความทนทาน (Robustness) ความถูกต้องแม่นยำในการทำงาน (Accuracy) เป็นต้น เพื่อให้ครอบคลุมคุณภาพในการให้บริการของเว็บเซอร์วิซให้มากขึ้น

6.3.3 เพิ่มความถูกต้องแม่นยำในการจับคู่ซอฟต์แวร์ที่ติดตั้งอยู่ในเซิร์ฟเวอร์ที่ให้บริการเว็บเซอร์วิซกับรายการซีวีอีเพื่อให้มีความเบี่ยงเบนน้อยที่สุด อาจมีการกำหนดกฎเกณฑ์อื่น ๆ ในการพิจารณาจับคู่เพื่อให้ได้รายการซีวีอีที่สอดคล้องกับจุดอ่อนของซอฟต์แวร์ที่มีอยู่ในเครื่อง เช่น ถ้าในเครื่องเซิร์ฟเวอร์ลงส่วนแก้ไข (Patch) ที่แก้ไขซีวีอีตัวนี้ไว้แล้ว ก็ไม่ควรนับว่าซีวีอีตัวนี้เป็นจุดอ่อนของเครื่องอีกต่อไป เป็นต้น

6.3.4 เพิ่มตัววัดอื่น ๆ ในแง่มุมของความมั่นคง เช่นเพิ่มตัววัดที่เกี่ยวข้องกับการจัดการเกี่ยวกับความลับของข้อมูลในการติดต่อกับเว็บเซอร์วิซ (Web Service Confidentiality) ตัววัดเกี่ยวกับการจัดการการเข้าถึงของข้อมูลในเว็บเซอร์วิซ (Web Service Access Control) เป็นต้น เพื่อให้ครอบคลุมการพิจารณาคุณภาพในการให้บริการในแง่ของความมั่นคงได้มากขึ้น

6.3.5 เนื่องจากเครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้น อาจจะมีผลกระทบต่อประสิทธิภาพการทำงานของเครื่องที่ให้บริการเว็บเซอร์วิซ ซึ่งควรจะมีการหาแนวทางหรือวิธีที่จะลดผลกระทบต่อประสิทธิภาพการทำงานของเครื่องที่ให้บริการเว็บเซอร์วิซเมื่อมีการติดตั้งเครื่องมือในการเฝ้าสังเกตนี้ เช่นมีการปรับแต่งค่าต่างๆในโครงแบบที่อยู่ในหน้า Extension Metrics Configuration ในแอดมินคอนโซล เพื่อให้ได้ค่าที่เหมาะสม (Optimal Value) ที่ทำให้เครื่องมือในการเฝ้าสังเกตส่งผลกระทบต่อการทำงานของเครื่องที่ให้บริการเว็บเซอร์วิซให้น้อยที่สุด เป็นต้น

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

รายการอ้างอิง

[1] Zeng, L.; Benatallah, B.; Ngu, A. H.; Dumas, M.; Kalagnanam, J.; and Chang, H. QoS-Aware Middleware for Web Services Composition. IEEE Transactions on Software Engineering, 30, 5 (May 2004): 311-327.

[2] Cardoso, J.; Sheth, A.; and Miller, J. Workflow Quality of Service. Proceedings of International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference, 2002, pp. 303-311.

[3] Ran, S. A Model for Web Services Discovery with QoS. ACM SIGEcom Exchanges, 4, 1 (Spring 2003): 1-10.

[4] Mani, A.; and Nagarajan, A. Understanding quality of service for Web services. [Online]. Available from: http://www.ibm.com/developerworks/library/ws-quality.html [2007, Feb 01]

[5] Soapknox. [Online]. Available from: http://www.soapknox.com/index.html [2007, Feb 01]

[6] WebInject. [Online]. Available from: http://www.webinject.org/ [2007, Feb 01]

[7] Application Server and GlassFish Project. [Online]. Available from: https://glassfish.dev.java.net/ [2007, Feb 01]

[8] ManageEngine Application Manager. [Online]. Available from: http://manageengine.adventnet.com/products/applications_manager/webservice-monitoring.html [2007, Feb 01]

[9] Microsoft Baseline Security Analyzer. [Online]. Available from: (http://www.microsoft.com/technet/security/tools/mbsahome.mspx) [2007, Feb 01]

[10] Sandcat. [Online]. Available from: http://www.syhunt.com/section.php?id=sandcat [2007, Feb 01]

[11] Understanding JMX Technology. [Online]. Available from: http://java.sun.com/developer/EJTechTips/2005/tt0222.html [2007, Feb 01]

[12] CVE: Common Vulnerabilities and Exposure. [Online]. Available from: http://www.cve.mitre.org/ [2007, Feb 01]

[13] NVD: National Vulnerability Database. [Online]. Available from: http://nvd.nist.gov/ [2007, Feb 01]

[14] CVSS: Common Vulnerability Scoring System version 2.0. [Online]. Available from: http://www.first.org/cvss/cvss-guide.html [2007, Feb 01]

[15] Kalepu, S.; Krishnaswamy, S.; and Loke, S. W. Verity: A QoS Metric for Selecting Web Services and Providers. Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops, December 2003, pp. 131-139.

[16] Wu, B. Y.; Chi, C. H.; and Xu, S. Service Selection Model Based on QoS Reference Vector. Proceedings of IEEE Congress on Services, Salt Lake City, Utah, 9-13 July 2007, pp. 270-277.

[17] Zeng, L.; Lei, H.; and Chang, H. Monitoring the QoS for Web Services. Proceedings of Internal Conference on Service-Oriented Computing (ICSOC 2007), pp. 132-144. LNCS 4749, 2007.

[18] Jurca, R.; Faltings, B.; and Binder, W. Reliable QoS Monitoring Based on Client Feedback. Proceedings of 16th International Conference on World Wide Web, Banff, Alberta, Canada, 2007, pp. 1003-1012.

[19] Baresi, L.; Ghezzi, C.; and Guinea, S. Smart Monitors for Composed Services. Proceedings of 2nd International Conference on Service-Oriented Computing, 15-19 November 2004, pp. 193-202.

[20] Rosenberg, F.; Platzer, C.; and Dustdar, S. Bootstrapping Performance and Dependability Attributes of Web Services, Proceedings of the IEEE International Conference on Web Services (ICWS'06), Chicago, USA, 18-22 September 2006, pp. 205-212.

[21] ปฏิญญา จันฤาชัย. เครื่องมือช่วยบริหารความปลอดภัยโดยใช้ฐานข้อมูลจุดอ่อนระบบเปิด. วิทยานิพนธ์ปริญญามหาบัณฑิต, ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2546.

[22] Ouzzani, M.; and Bouguettaya, A. Efficient Access to Web Services. IEEE Internet Computing, 8, 2 (March-April 2004): pp. 34-44.

[23] JavaServer Faces Technology. [Online]. Available from: http://java.sun.com/javaee/javaserverfaces/ [2009, Feb 01]

ภาคผนวก

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ก

วิธีการติดตั้งเครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้น

ในภาคผนวก ก จะกล่าวถึงวิธีในการติดตั้งเครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้น ลงในระบบปฏิบัติการวินโดวส์ ซึ่งเครื่องมือในการเฝ้าสังเกตและจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์จะอยู่ในไฟล์ที่ชื่อว่า JavaAppServerWithExMonitorTool.zip โดยขั้นตอนในการติดตั้งมีดังต่อไปนี้

1. แตกไฟล์ JavaAppServerWithExMonitorTool.zip ลงในไดเรกทอรีที่เตรียมไว้ติดตั้งจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ซึ่งในตัวอย่างนี้จะเป็น D:/workspace

2. หลังจากที่แตกไฟล์เสร็จเรียบร้อยแล้ว เมื่อเข้าไปยังไดเรกทอรี D:/workspace เราจะพบโฟลเดอร์ทั้งหมด 3 โฟลเดอร์ ซึ่งได้แก่
   1) โฟลเดอร์ glassfish ซึ่งเป็นโฟลเดอร์ที่เก็บรหัสต้นฉบับ (Source Code) ของเครื่องมือในการเฝ้าสังเกตที่ได้พัฒนาขึ้น ซึ่งเมื่อแปลโปรแกรม (Compile) รหัสต้นฉบับเหล่านี้แล้วจะถูกแปลงเป็นไฟล์ประมวลผล (Execution File) แล้วถูกเก็บลงในโฟลเดอร์ publish
   2) โฟลเดอร์ glassfish_dependencies เป็นโฟลเดอร์ที่เก็บไฟล์ตั้งต้นของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์เอาไว้ ซึ่งจะถูกแปลโปรแกรมรวมกับรหัสต้นฉบับในโฟลเดอร์ glassfish เพื่อแปลงเป็นไฟล์ประมวลผลแล้วถูกเก็บลงในโฟลเดอร์ publish ต่อไป ซึ่งโฟลเดอร์ glassfish_dependencies จะช่วยให้นักพัฒนาสามารถดึงรหัสต้นฉบับเฉพาะที่ตนเองต้องการใช้มาทำการแก้ไข โดยไม่จำเป็นต้องดึงรหัสต้นฉบับทั้งหมดของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์มาแปลโปรแกรม
   3) โฟลเดอร์ publish เป็นโฟลเดอร์ที่เก็บไฟล์ประมวลผลทั้งหมดของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ ดังนั้นการทำงานของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ทั้งหมดจะอยู่ที่โฟลเดอร์นี้

3. เปิด วินโดวส์คอมมานด์พรอมท์ (Windows Command Prompt) จากวินโดวส์ เมนู หลังจากนั้นใช้คำสั่ง cd ตามด้วยตำแหน่งของโฟลเดอร์ที่เก็บพวกไฟล์ประมวลผล ดังตัวอย่างข้างล่าง

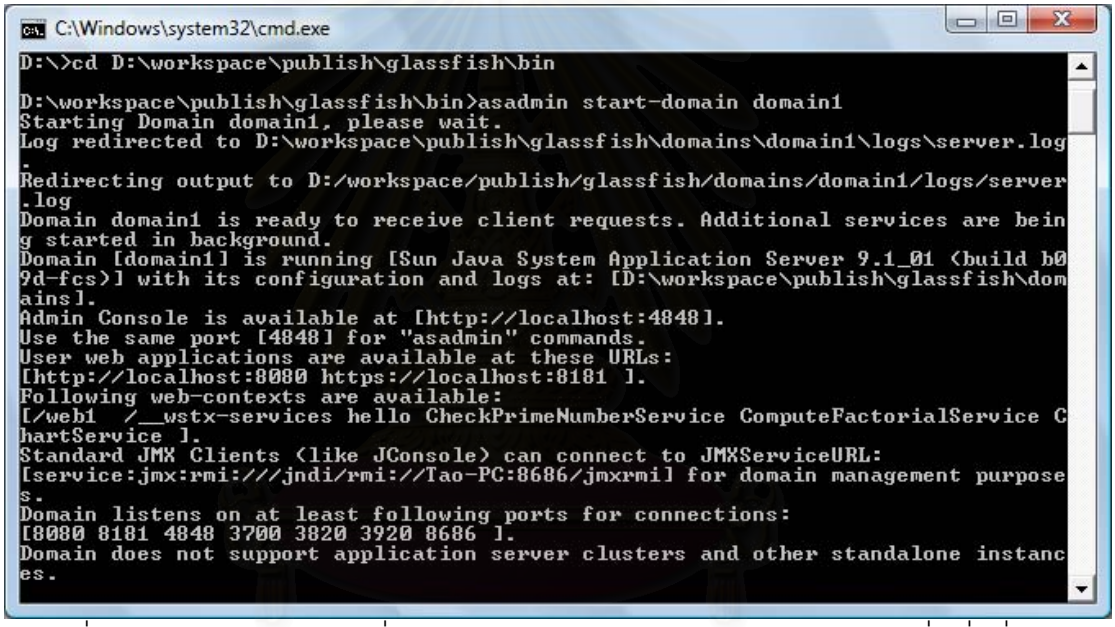
```
cd  <base directory>\publish\glassfish\bin
```

โดย <base directory> คือตำแหน่งของไดเรกทอรีที่ได้ทำการแตกไฟล์ JavaAppServerWithExMonitorTool.zip ไปไว้ที่ไดเรกทอรีนั้น

4. พิมพ์คำสั่งต่อไปนี้เพื่อเริ่มต้นการทำงานของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์

```
asadmin start-domain domain1
```

หลังจากนั้นจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์จะแสดงรายละเอียดต่าง ๆ เกี่ยวกับการทำงานของตัวระบบเช่น ตำแหน่งของล็อกไฟล์ ยูอาร์แอลที่ใช้ในการเรียกเว็บแอพพลิเคชัน ยูอาร์แอลในการเรียกแอดมินคอนโซลเป็นต้น ดังรูปที่ ก-1



รูปที่ ก-1 รายละเอียดต่าง ๆ เกี่ยวกับการทำงานของตัวระบบหลังจากพิมพ์คำสั่งเพื่อเริ่มการทำงานของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์

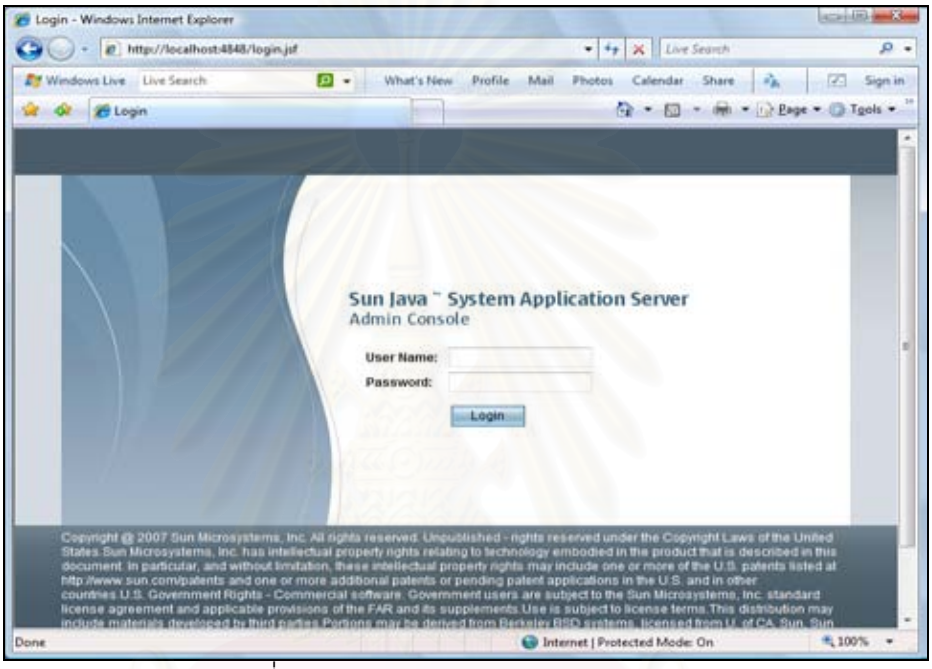
5. ในกรณีที่ต้องการจบการทำงานของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ ให้ใช้คำสั่ง cd ในวินโดวส์คอมมานด์พรอมท์ ไปที่ตำแหน่งของโฟลเดอร์เดิมที่เก็บพวกไฟล์ประมวลผล แล้วพิมพ์คำสั่งต่อไปนี้เพื่อจบการทำงานของจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์

```
asadmin stop-domain domain1
```

เราสามารถตรวจสอบว่าจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ติดตั้งได้สมบูรณ์หรือไม่ โดยการเปิดอินเทอร์เน็ตเอ็กซ์โพลเรอร์ (Internet Explorer) แล้วพิมพ์ยูอาร์แอลต่อไปนี้ลงไป

http://locahost:4848/

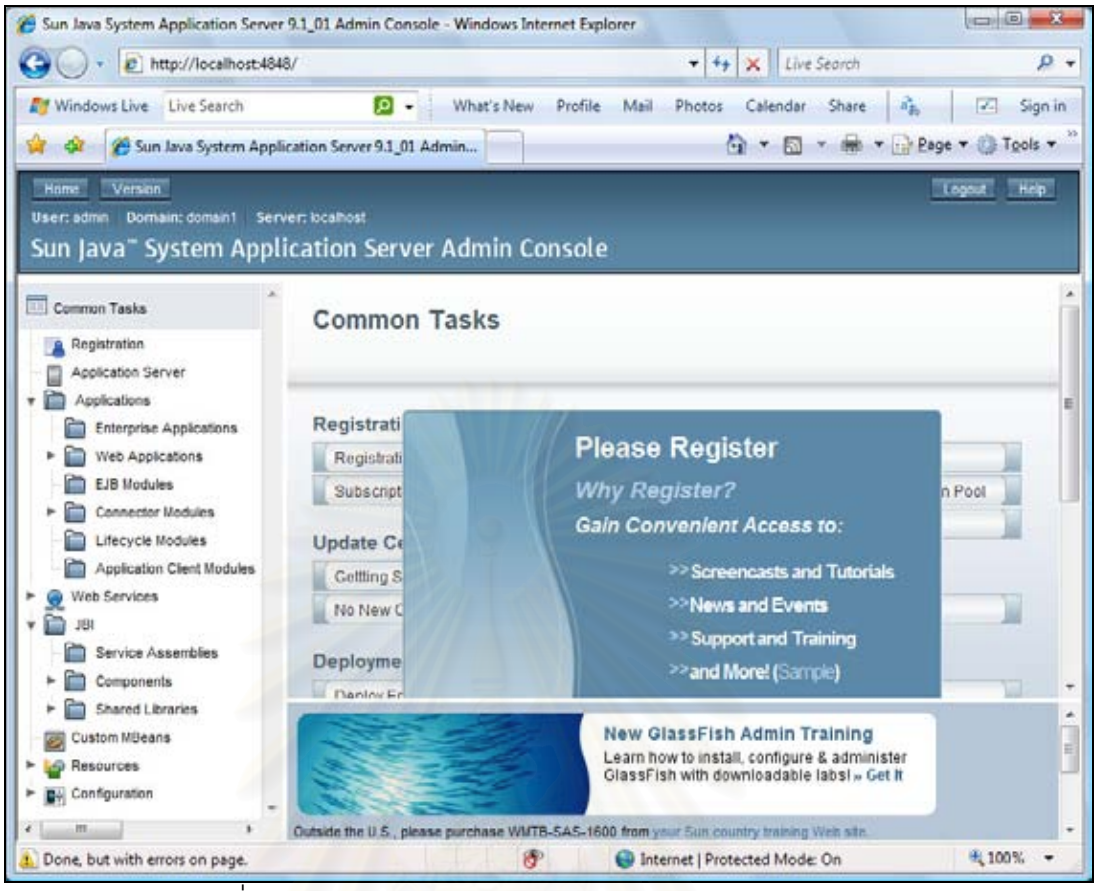
ยูอาร์แอลที่พิมพ์ไปนั้น เป็นยูอาร์แอลของแอดมินคอนโซล ซึ่งเริ่มแรกจะแสดงหน้าจอเพื่อให้ผู้ใช้ล็อกอินเข้าทำการใช้งานดังรูปที่ ก-2



รูปที่ ก-2 หน้าล็อกอินของแอดมินคอนโซล

ล็อกอินเข้าสู่ระบบโดยใช้ยูสเซอร์เนม (User Name) เป็น admin และ พาสเวิร์ด (Password) เป็น adminadmin ซึ่งถ้าติดตั้งจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์สำเร็จจะแสดงหน้าจอดังรูปที่ ก-3

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย



รูปที่ ก-3 หน้าจอของแอดมินคอนโซลหลังจากล็อกอินได้สำเร็จ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ข

วิธีการติดตั้งเว็บเซอร์วิซในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์

1. สร้างเว็บเซอร์วิซที่จะใช้ติดตั้งลงในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ โดยมีขั้นตอนในการสร้างดังต่อไปนี้

   1) อิมพอร์ต (Import) คลาส javax.jws.WebService

   2) ใส่สัญลักษณ์ @WebService ที่หน้าคลาสที่ต้องการให้เป็นเว็บเซอร์วิซ ซึ่งในตัวอย่างนี้จะสร้างจาวาคลาสที่ชื่อว่า ComputeFactorialService ซึ่งมีหนึ่งฟังก์ชัน ชื่อ computeFactorial ซึ่งจะรับข้อมูลขาเข้าเป็นตัวเลขจำนวนเต็ม แล้วคำนวณค่าเอ็นแฟคทอเรียล (N Factorial) โดย N คือค่าข้อมูลขาเข้าที่รับเข้าไป ซึ่งรหัสต้นฉบับของคลาส ComputeFactorialService ทั้งหมดสามารถแสดงได้ดังรูปที่ ข-1

```
package server;
import javax.jws.WebService;

@WebService
public class ComputeFactorialService{
   /*** @param nunmber (Integer value)
        @return result (Integer value)  */
   public long computeFactorial(long number) {
                long result = 1;
                for (int i = 1; i <= number; i++){
                        result = result * i;
                }

                return result;
   }
}
```

รูปที่ ข-1 รหัสต้นฉบับของคลาส ComputeFactorialService

   3) บันทึกไฟล์ลงในชื่อ ComputeFactorialService.java

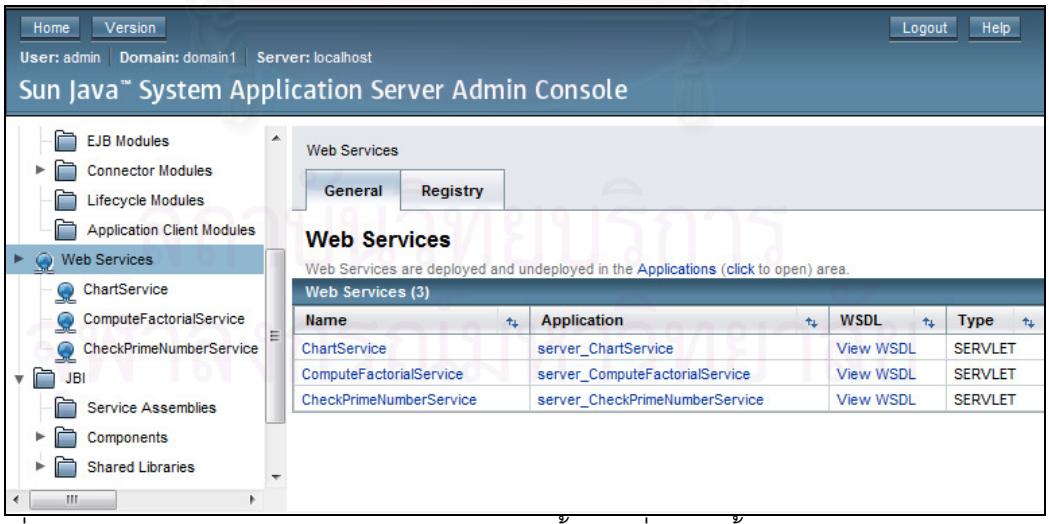
4) แปลโปรแกรม โดยใช้คำสั่งดังต่อไปนี้

```
javac -classpath <base directory>/publish/glassfish/lib/javaee.jar -d <base directory>/publish/glassfish/domains/domain1/autodeploy ComputeFactorialService.java
```

โดย <base directory> คือตำแหน่งของไดเรกทอรีที่ได้ทำการแตกไฟล์ JavaAppServerWithExMonitorTool.zip ไปไว้ที่ไดเรกทอรีนั้น

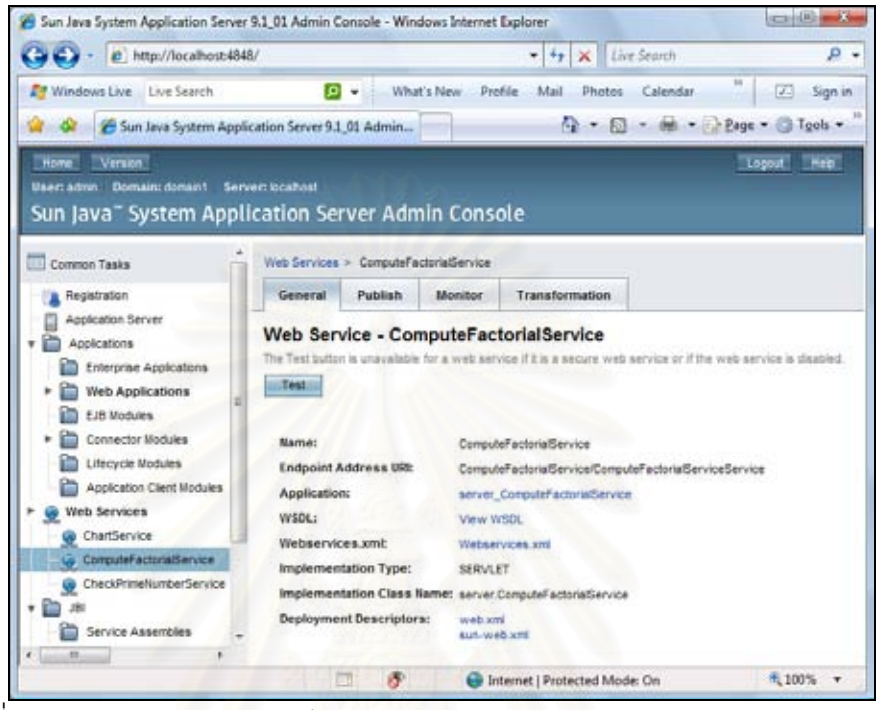
จากคำสั่งในข้อ 4) จาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์จะแปลโปรแกรมและติดตั้งคลาสที่ชื่อว่า ComputeFactorialService ให้เป็นเหมือนเว็บเซอร์วิซ โดยมีหนึ่งการปฏิบัติการ (Operation) ชื่อ computeFactorial

2. เปิดอินเทอร์เน็ตเอ็กซ์โพลเรอร์ แล้วพิมพ์ยูอาร์แอล http://localhost:4848/ เพื่อไปยังแอดมินคอนโซล
3. ล็อกอินเข้าสู่ระบบโดยใช้ยูสเซอร์เนมเป็น admin และ พาสเวิร์ดเป็น adminadmin
4. เมื่อล็อกอินสำเร็จแล้ว คลิกเลือกเมนู Web Services จากเมนูต้นไม้ (Tree Menu) ที่อยู่ทางด้านซ้ายมือ ซึ่งจะแสดงรายละเอียดของเว็บเซอร์วิซทั้งหมดที่ถูกติดตั้งในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ ซึ่งเว็บเซอร์วิซ ComputeFactorialService ก็จะแสดงอยู่ในนั้นด้วย ดังรูปที่ ข-2

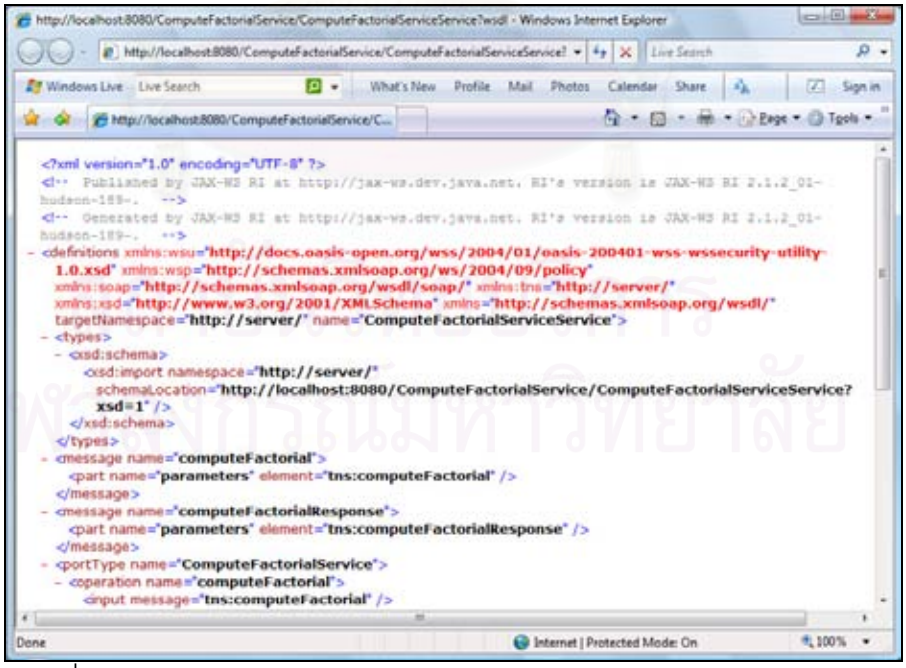


รูปที่ ข-2 หน้าจอแสดงรายละเอียดของเว็บเซอร์วิซทั้งหมดที่ถูกติดตั้งในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์

5. คลิกที่ชื่อของเว็บเซอร์วิซ ComputeFactorialService จากเมนูต้นไม้ ซึ่งจะแสดงรายละเอียดต่าง ๆ ของเว็บเซอร์วิซ ComputeFactorialService รวมถึงสามารถดูรายละเอียดของไฟล์วิสเดิล (WSDL) ของเว็บเซอร์วิซโดยคลิกที่ View WSDL ดังรูปที่ ข-3 และ ข-4

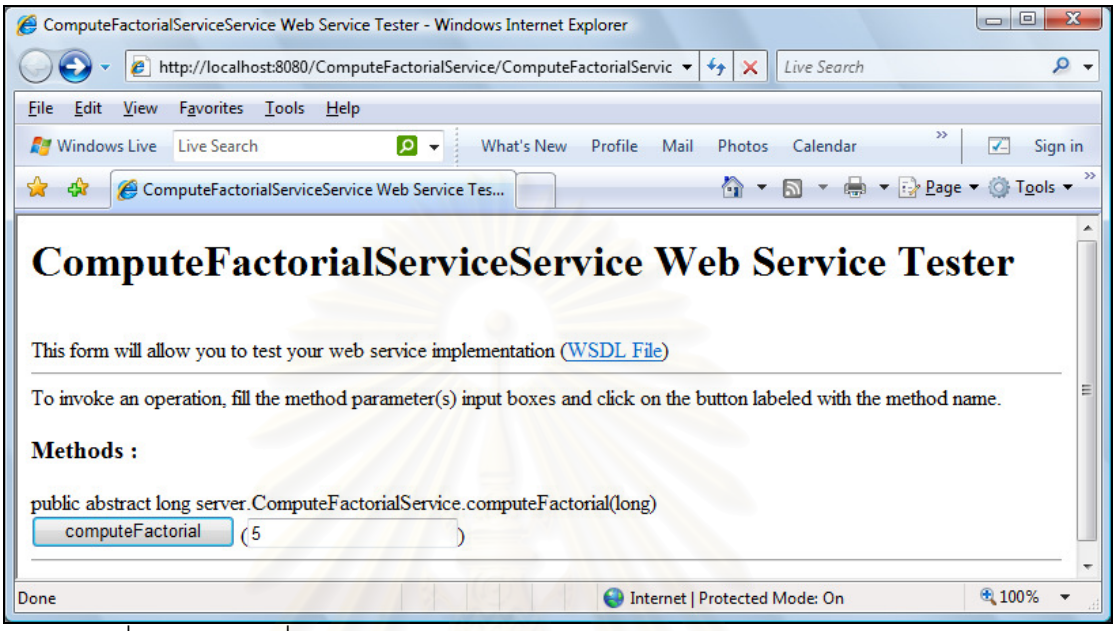


รูปที่ ข-3 หน้าจอแสดงรายละเอียดต่าง ๆ ของเว็บเซอร์วิซ ComputeFactorialService

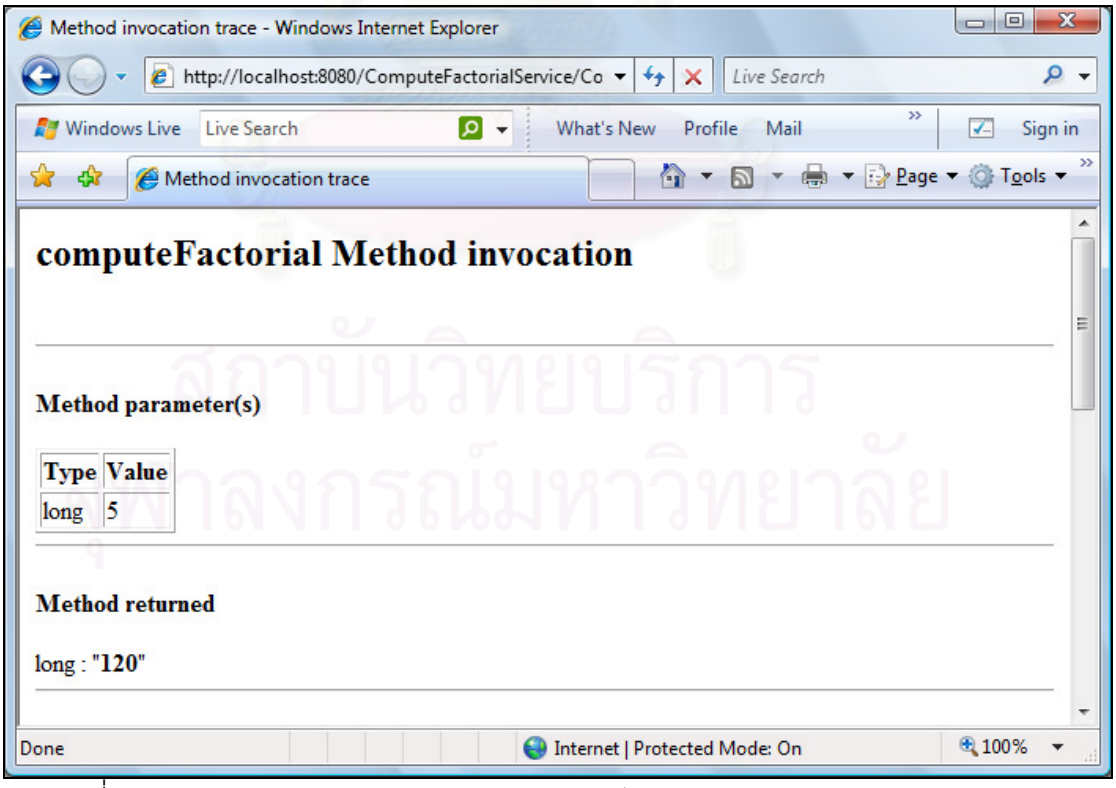


รูปที่ ข-4 หน้าจอแสดงรายละเอียดของไฟล์วิสเดิล (WSDL) ของเว็บเซอร์วิซ

6. นอกจากนี้ เราสามารถทดสอบการทำงานของเว็บเซอร์วิซที่ได้ติดตั้งในจาวาซิสเต็มแอพพลิเคชันเซิร์ฟเวอร์ โดยคลิกที่ปุ่ม Test แล้วลองใส่ค่าลงไป เพื่อทดสอบการทำงานของเว็บเซอร์วิซ ComputeFactorialService ดังรูปที่ ข-5 และ ข-6



รูปที่ ข-5 หน้าจอที่ใช้ทดสอบการทำงานของเว็บเซอร์วิซ ComputeFactorialService



รูปที่ ข-6 ผลลัพธ์จากการทดสอบการทำงานของเว็บเซอร์วิซ ComputeFactorialService

ภาคผนวก ค

ผลงานตีพิมพ์

ส่วนหนึ่งของงานวิจัยนี้ได้รับการตีพิมพ์ในหนังสือรวมบทความในการประชุมวิชาการนานาชาติ 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD2008) ณ โรงแรม Hilton Phuket Arcadia Resort and Spa จังหวัดภูเก็ต ระหว่างวันที่ 6-8 สิงหาคม พ.ศ. 2551 ในบทความเรื่อง Enhancing Service-Side QoS Monitoring for Web Services หน้า 765-770 จัดทำและเผยแพร่โดย IEEE Computer Society, Washington, DC

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing

Enhancing Service-Side QoS Monitoring for Web Services

Natee Artaiam
Department of Computer Engineering
Chulalongkorn University
Phyathai Road, Pathumwan,
Bangkok 10330 Thailand
natee.a@student.chula.ac.th

Twittie Senivongse
Department of Computer Engineering
Chulalongkorn University
Phyathai Road, Pathumwan,
Bangkok 10330 Thailand
twittie.s@chula.ac.th

Abstract

Quality of service (QoS) is adopted widely in Web services research and practice for various purposes such as for metadata publishing and operational management of Web services. Nonetheless, current tools for monitoring and gathering QoS information support only a limited number of QoS attributes (e.g. response time, successful execution rate, availability) and therefore can neither well keep up to the QoS models proposed in several researches nor to the requirements in practical use. This paper reviews a QoS model which covers various dimensions of service quality (i.e. availability, accessibility, performance, reliability, security, and regulatory) and proposes metrics to enhance QoS measurement on the service side. A monitoring tool is designed and developed as an extension to Web services monitoring feature of Java System Application Server under Sun's GlassFish project.

1. Introduction

Quality of service (QoS) has been adopted widely in Web services research and practice. Service consumers differentiate Web services of the same functionalities by their quality during service discovery and composition. Service providers concentrate on the QoS when publishing service metadata and monitoring the quality in order to manage their service provision. Several publications [1-4] define different QoS models but the most common QoS attributes include time, cost, availability, reliability, reputation, and security of services. Among those models, some give only the definitions of the QoS while others define also the accompanying metrics for measuring such quality aspects. In some cases, the QoS attributes of the same name have distinct meanings under different models, e.g. time in one model refers to response time measured at the client side whereas it means execution time measured at the service side in another model.

Web services QoS is usually determined by collecting service usage data in order to compute the degree of quality based on quality metrics. Many Web service monitoring tools take slightly different approaches to collect Web services usage and calculate the QoS; some focus on service performance while some on security aspect. Despite the availability of these tools, only a limited number of QoS attributes are supported and some attributes (e.g. security) are determined qualitatively. This paper aims to enhance service-side QoS monitoring for Web services by considering more quality aspects and trying to represent service quality in a more quantitative manner. We consider six QoS attributes in total – availability, accessibility, performance, reliability, security, and regulatory – and present a metric for each quality aspect. A Web services monitoring extension is developed for Java System Application Server.

The rest of the paper is organized as follows. Section 2 discusses related work and Section 3 presents the QoS model and accompanying metrics that are used. The design and implementation of our Web services monitoring tool are discussed in Section 4. Section 5 draws some conclusions on the paper.

2. Related Work

We discuss related work from three views: existing QoS monitoring tools, QoS modeling, and service monitoring mechanisms.

On QoS monitoring tools, several of them are available for different run-time Web services environments, but not many QoS attributes are

978-0-7695-3263-9/08 $25.00 © 2008 IEEE
DOI 10.1109/SNPD.2008.157

IEEE computer society

supported. For example, SoapKnox [5] is a plug-in tool for service-side monitoring available in several platforms (e.g. WSO2, Axis, .NET). It can log Web service invocations and calculate response time and successful execution rate. SOAP content, including security-related information in the header, can also be viewed. WebInject [6] is a client-side monitoring tool that can send requests to Web applications or Web services and analyze from responses the response time and success and fault request counts. Java System Application Server [7] is the open source software developed by Sun's GlassFish project with Java EE 5.0. It is the application server with an AppServer Management Extension (AMX) that supports service-side monitoring of response time, throughput (i.e. the number of requests per second), and success and fault request counts for Web services. SOAP content, including security-related information in the header, can also be viewed. ManageEngine Application Manager [8] is a client-side monitoring tool that can send requests to Web services and analyze from responses the response time and availability (i.e. downtime and uptime period). Microsoft Baseline Security Analyzer [9] is a service-side tool that can gather and analyze security vulnerabilities in Web server machines that host Web services. It identifies vulnerabilities such as Windows, IIS, and SQL administrative vulnerabilities, weak passwords, and missing security updates. The degree of vulnerability is determined on a qualitative scale, e.g. best practice, passed, failed (but vulnerability is non-critical), failed (and vulnerability is critical). Sandcat [10] is a service-side tool that also can scan Web sites and Web servers that host Web services for security vulnerabilities. It lists vulnerability names and their impacts (i.e. critical and non-critical).

On QoS modeling, substantial research works have proposed QoS models to be used in service discovery and composition; however, they use these models merely for simulation of their mechanisms and do not concern how the QoS values are actually obtained. For example, a QoS model for Web services in [1] captures execution price, execution duration, reputation, successful execution rate, and availability. The values of most attributes can be analyzed from service invocations and responses, except for execution price which has to be specified by service providers, and service reputation which is ranked subjectively by service consumers. In [2], time, cost, reliability, and fidelity (i.e. how well a service is rendered and meets consumer specifications) constitute the QoS model. The paper [3] addresses a quality-aware UDDI for Web services discovery based on quality such as availability, reliability, throughput, response time, latency, security, robustness, accuracy, and integrity of Web services. A quality metric called verity is introduced in [11] to contribute to service reputation. Verity refers to the degree of variance in the compliance levels of a service provider with the QoS laid out in the service level agreement. The work in [12] proposes a QoS reference vector to model price, time, reliability, trust (i.e. subjective ranking by consumers), and security (on some unknown scale, 1-5).

On service monitoring mechanisms, the paper [13] proposes a rich set of QoS metrics and a client-side monitoring tool that supports several kinds of performance and dependability quality. Other research works focus on the overall monitoring architectures rather than the provision of a richer QoS support. For example, the work in [14] presents a high-performance QoS monitoring system with a service operations observation model that defines IT- and business-level metrics and associated evaluation formulas. The paper [15] addresses that, without proper penalties, service providers have strong incentives to deviate from the advertized QoS, causing losses to the clients. Therefore, the focus is on a QoS monitoring and reputation mechanism that collects ratings from the clients and computes the actual quality delivered. In [16], implementations of the monitors for BPEL composite services are discussed. The monitors can watch for timeout, run-time errors, and pre-/post-condition violation on BPEL services and act upon some predefined exception handling.

3. QoS Model for Web Services Monitoring

As we focus on a QoS model with a wide range of quality dimensions that can be monitored and determined quantitatively on the service side, six QoS attributes in [4] are adopted as described below. (Note that the integrity attribute in [4] is not included here as it relates to Web services in transactions.) We also propose a metric for each quality dimension from service providers' perspective. This QoS model can be adopted by service providers for publishing service metadata and maintaining the quality of service provision. The quality attributes are:

1. **Availability** is the quality aspect of whether a Web service is present or ready for immediate use. It represents the probability that a service is available (i.e. up and running) [1, 3, 13]. The metric for availability is:

$$A_v(s) = T_a(s)/t$$

where $A_v(s)$ is the availability value of Web service s

$T_s(s)$ is the total amount of time (in seconds) that s has been up during the observation period

t is the total amount of the observation period (in seconds).

2. **Accessibility** is the quality aspect of a Web service that represents the degree it is capable of serving a request. There could be situations when a Web service is available but not accessible due to high volume of requests. We propose a metric for accessibility as follows:

$$A_c(s) = K(s)/t$$

where $A_c(s)$ is the accessibility value of Web service s

K(s) is the total number of requests for which the service s has successfully delivered valid responses within an expected time frame

t is the total amount of the observation period (in seconds)

3. **Performance** is the quality aspect of a Web service which can be measured in terms of throughput (i.e. the number of requests served at a given time period) and latency (e.g. round-trip time between sending a request and receiving a response). Since our focus is on service-side measurement, we represent performance by execution time [13] which refers to the time the service takes to handle the request (i.e. unwrap the XML structure of the request, process the request, and wrap the answer into a SOAP response). The metric for execution time is:

$$T_e(s) = \left(\sum_{i=1}^{N(s)} \Delta Ti \right) / N(s)$$

where $T_e(s)$ is the average execution time of Web service s

i is the i^{th} request that is sent to the service s

N(s) is the total number of requests to the service s during an observation period

ΔTi is the execution time (in milliseconds) of the i^{th} request which is computed by the difference between departure timestamp of the response and arrival timestamp of the request at the service side

4. **Reliability** is the quality aspect of a Web service that represents the degree of being capable of maintaining the service and service quality. We adopt the metric for successful execution rate in [1] to represent reliability. It is the probability that a service request is correctly responded with a valid response within an expected time frame. It is computed based on past invocations to the service:

$$R_{rel}(s) = K(s)/N(s)$$

where $R_{rel}(s)$ is the reliability value of Web service s

K(s) is the total number of requests for which the service s has successfully delivered valid responses within an expected time frame

N(s) is the total number of requests to the service s during an observation period

5. **Security** is the quality aspect of a Web service of providing confidentiality and non-repudiation by authenticating the parties involved, encrypting messages, and providing access control. However, it is difficult to automatically and quantitatively measure such aspects. In this paper, we represent security quality by impact scores of security vulnerabilities that are present in Web services operational environment. We adapt from the approach of [17] to scan the machine that hosts the service for software information, and look up the National Vulnerability Database (NVD) [18] for the Common Vulnerabilities and Exposures list (CVE list) for such software [19]. CVE is a dictionary that provides common names for publicly known information security vulnerabilities and exposures, while NVD is a comprehensive database that integrates all US Government publicly available vulnerability resources. NVD supports a quantitative scoring model called Common Vulnerability Scoring System (CVSS) [20] which gives impact scores for software vulnerabilities. We assume that vulnerabilities of any software on the service-hosting machine should affect the service operation if they are attacked, whether or not the software relates directly to the Web services. We propose a metric for security based on an average CVSS base score (i.e. score that does not consider temporal and environmental effects on the impact of vulnerabilities) of all CVEs found in a Web service-hosting machine:

$$S_v(s) = \left(\sum_{i=1}^{n} CVSS(CVEi) \right) / n$$

where $S_v(s)$ is the vulnerability impact value of Web service s

$CVSS(CVEi)$ is the CVSS base score of the i^{th} CVE (ranging from 0 to 10)

n is the total number of CVEs found in the service-hosting machine

6. **Regulatory** is the quality aspect of a Web service in conformance with rules, law, compliance with standards, and established service level agreement. A metric in [21] is adopted to measure the gap between the published and the actually delivered QoS. The gap can reflect how service quality fluctuates; a positive/negative gap means the service delivers better/poorer quality than what has been published:

$$R_{reg}(s) = \sum_{Qa \in pos} (dQi - pQi) + \sum_{Qa \in neg} (\frac{1}{dQi} - \frac{1}{pQi})$$

where $R_{reg}(s)$ is the regulatory value of Web service s

dQi is the delivered value of the i^{th} QoS metric

pQi is the published value of the i^{th} QoS metric

pos is a set of positive QoS attributes to maximize, i.e. availability, accessibility, and reliability

neg is a set of negative QoS attributes to minimize, i.e. execution time and vulnerability impact.

To compute the regulatory quality, values of some QoS attributes will need to be normalized first. For the accessibility attribute, the value can be greater than 1 and therefore has to be normalized to 0-1 by dividing the delivered value and the published value with the published value. For the vulnerability impact attribute, the value can range from 0 to 10, and also has to be normalized to 1-11 by adding 1 to the published value and the delivered value.

4. Design and Implementation of Service Monitoring Extension

We realize the monitoring of QoS dimensions above through a monitoring extension of Java system application server developed in Java EE 5.0 [7]. This open source application server is chosen because service providers can deploy their Web services and get benefits from our monitoring extension without having to install additional packages. We can build on its current monitoring supports, i.e. response time (which corresponds to our execution time attribute), throughput, and success and fault request counts.

Figure 1 shows the extension (in shaded boxes) to the monitoring capability of Java system application server. The overall architecture comprises the following:

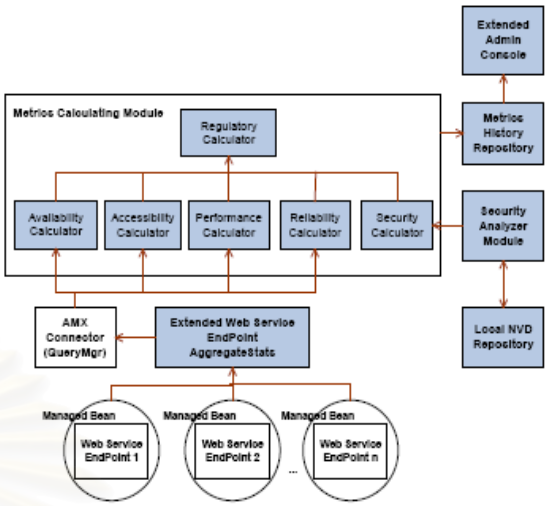


Figure 1. Design of service monitoring extension

1. **Managed Bean** or MBean is a concept of a managed resource in Java Management Extension (JMX) on which the monitoring is based. It is a Java object that wraps a resource, i.e. a Web service, and provides activities data captured at the Web service endpoint, i.e. messages, message timestamps, failure status. It is managed by a JMX agent.

2. **Extended Web Service Endpoint AggregateStats** is an extension to the existing Web Service Endpoint AggregateStats which is an interface that provides access to monitored statistical data gathered from managed beans. We extend it with the ability to provide data derived from heartbeat messages that are sent to monitor service health (availability).

3. **AMX Connector (Query Mgr)** is an object of the AppServer Management Extension (AMX) that is built on top of JMX to query statistical data through the Extended Web Service Endpoint AggregateStats interface.

4. **Security Analyzer Module** and **Local NVD Repository** together provide vulnerability impact information of Web services. The Security Analyzer Module first scans a Web service-hosting machine for a list of installed software. As our implementation supports Windows platform, the list of names and versions of software is compiled from the Windows Registry keys. This list is compared with CVE information from the Local NVD Repository in order to identify relevant vulnerabilities. Note that the Local NVD Repository is effectively the local copy of XML files containing CVE information and accompanying CVSS scores. It is updated with the NVD Web site when vulnerability impact scores are to be analyzed. For an organization that hosts Web services in closed

internal environments, the Local NVD Repository will be updated when the Internet connection can be made.

5. **Metrics Calculating Module** retrieves monitored data from the AMX Connector and Security Analyzer Module and calculates QoS metrics using corresponding calculating modules. Note that the regulatory value is based on other monitored QoS values and the published QoS level that is maintained by the application server.

6. **Metrics History Repository** is an XML file to which all computed QoS values can be exported. It maintains the history of Web services QoS to enable further analysis for service provision management, e.g. to adjust published QoS, improve machine capacity, or harden service environment. It can be configured to maintain the history for a certain period.

7. **Extended Admin Console** is an extension to the existing Admin Console (i.e. admin-gui module) to report all QoS information that is computed. It can show also the QoS history of Web services. Figure 2 shows the Extension Metrics tab, displaying all supported QoS metrics, of a ChartService. In Figure 3, the Vulnerability Information tab lists the installed software with vulnerabilities and average scores.

5. Discussion and Conclusion

The contributions of this paper comprise a richer service-side QoS model and its realization as an extension to the QoS monitoring capability of Java system application server. Our approach enhances the status of Web services QoS monitoring which is currently supported in a limited fashion. The total of six QoS aspects are taken into account, and we either adopt existing quality metrics (i.e. availability, performance, and reliability metrics) or present new ones (i.e. accessibility, security, and regulatory metrics). See Table 1 for summary.

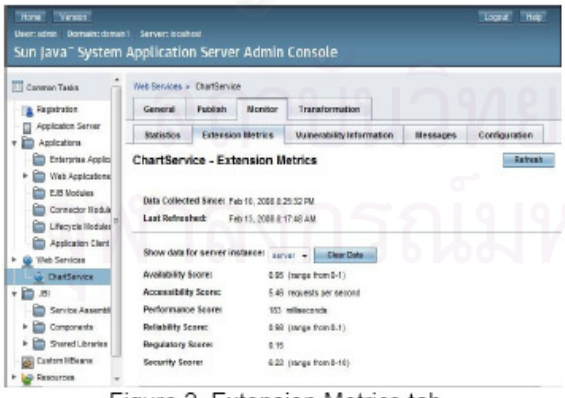


Figure 2. Extension Metrics tab

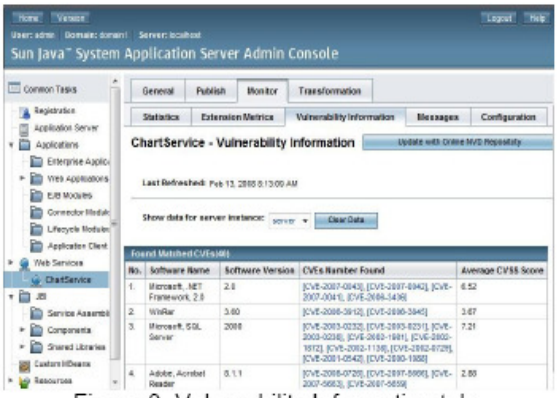


Figure 3. Vulnerability Information tab

Unlike client-side monitoring, service-side monitoring has a constraint such that only the services under control (i.e. hosted by the application server) can be monitored. That is, in our case, the monitoring extension cannot monitor other public services. However, it is necessary to provide such a service-side mechanism because service providers still require some QoS information that can be gathered only at the host machine in order to manage their services, e.g. execution time and vulnerability information.

The QoS model can be used as a guideline for service providers to publish their service quality and to concentrate in the management of provided services in order to keep up to the published quality level. Even though QoS aspects and metrics are presented, we do not claim that they constitute a complete QoS model. In other words, additional QoS aspects can be explored such as robustness, accuracy, and reputation. The presented metrics should not be considered sufficient for measuring the designated quality aspects either. For example, we may use the metric that measures the degree of assured and ordered message delivery to represent reliability quality. The vulnerability impact alone does not represent a sufficient measurement for security; we may also consider other means to quantitatively represent the degree of confidentiality and access control exercised by a service. We will explore further the standards such as WS-Security and WS-ReliableMessaging to enhance our approach.

The QoS has typically been associated at the network/server level rather than the user level. As a diversity of Web service applications is emerging, it is now necessary to relate the objective system quality to the consumers' subjective perception of quality. The approach such as in [11] improves subjective ranking of service reputation based on QoS compliance level and is interesting for the QoS model. We will also evaluate the overheads incurred by the monitoring extension in the future.

Table 1. Comparison of Web services monitoring tools

Tools	Availability	Accessibility	Performance	Reliability	Security	Regulatory
SoapKnox	N/A	N/A	Response time	Successful execution rate	Monitor SOAP header	N/A
WebInject	N/A	N/A	Response time	Success/Fault request counts	N/A	N/A
Java System Application Server	N/A	N/A	Response time, Throughput	Success/Fault request counts	Monitor SOAP header	N/A
ManageEngine Application Manager	Uptime, Downtime	N/A	Response time	N/A	N/A	N/A
Microsoft Baseline Security Analyzer	N/A	N/A	N/A	N/A	List of vulnerabilities and qualitative impacts	N/A
Sandcat	N/A	N/A	N/A	N/A	List of vulnerabilities and qualitative impacts	N/A
Our monitoring extension	$A_v(s)$	$A_c(s)$	$T_e(s)$	$R_{rel}(s)$	List of vulnerabilities and $S_v(s)$ scores	$R_{reg}(s)$

Acknowledgment

This research is part of the Engineering New Paradigm Software for Enterprises with Service-Oriented Architecture Project, supported by Software Industry Promotion Agency (Public Organization).

References

[1] L. Zeng, B. Benatallah, A.H.H Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition", *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, May 2004, pp. 311-327.

[2] J. Cardoso, A. Sheth, and J. Miller, "Workflow Quality of Service", *Proceedings of International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference*, 24-26 April 2002, pp. 303-311.

[3] S. Ran, "A Model for Web Services Discovery with QoS", *ACM SIGEcom Exchanges*, Vol.4, No.1, Spring 2003, pp.1-10.

[4] A. Mani and A. Nagarajan, "Understanding Quality of Service for Web Services", *IBM Developer Works Technical Paper*, January 2002, Available at : http://www-128.ibm.com/developerworks/library/ws-quality.html

[5] SoapKnox. Available at http://www.soapknox.com/ index. html

[6] WebInject. Available at http://www.webinject.org/

[7] Java System Application Server and GlassFish Project. Available at https://glassfish.dev.java.net/

[8] ManageEngine Application Manager. Available at http:// manageengine.adventnet.com/products/ applications_manager/webservice-monitoring.html

[9] Microsoft Baseline Security Analyzer. Available at http:// www.microsoft.com/technet/security/tools/ mbsahome.mspx

[10] Sandcat. Available at http://www.syhunt.com/ section.php?id=sandcat

[11] S. Kalepu, S. Krishnaswamy, and S. W. Loke, "Verity: A QoS Metric for Selecting Web Services and Providers", *Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops*, December 2003, pp. 131-139.

[12] B.Y. Wu, C.H. Chi and S. Xu, "Service Selection Model Based on QoS Reference Vector", *Proceedings of IEEE Congress on Services*, Salt Lake City, Utah, 9-13 July 2007, pp. 270-277.

[13] F. Rosenberg, C. Platzer, and S. Dustdar, "Bootstrapping Performance and Dependability Attributes of Web Services", *Proceedings of the IEEE International Conference on Web Services* (ICWS'06), Chicago, USA, 18-22 September 2006, pp. 205-212.

[14] L. Zeng, H. Lei, and H. Chang, "Monitoring the QoS for Web Services", *Proceedings of Internal Conference on Service-Oriented Computing (ICSOC 2007)*, LNCS 4749, 2007, pp. 132-144.

[15] R. Jurca, B. Faltings, and W. Binder, "Reliable QoS Monitoring Based on Client Feedback", *Proceedings of 16th International Conference on World Wide Web*, Banff, Alberta, Canada, 2007, pp. 1003-1012.

[16] L. Baresi, C. Ghezzi and S. Guinea, "Smart Monitors for Composed Services", *Proceedings of 2nd International Conference on Service-Oriented Computing*, 15-19 November 2004, pp. 193-202.

[17] P. Janluechai, "Security Administrative Tool Based On Open Source Vulnerability Database", Master Thesis, Department of Computer Engineering, Chulalongkorn University, 2004.

[18] NVD: National Vulnerability Database version 2.0. Available at http://nvd.nist.gov/

[19] CVE: Common Vulnerabilities and Exposures. Available at http://www.cve.mitre.org/

[20] CVSS: Common Vulnerability Scoring System version 2.0. Available at http://www.first.org/cvss/cvss-guide.html

[21] M. Ouzzani and A. Bouguettaya, "Efficient Access to Web Services", *IEEE Internet Computing*, Vol.8, No. 2, March-April 2004, pp. 34-44.

ประวัติผู้เขียนวิทยานิพนธ์

นายนที อาจเอี่ยม เกิดเมื่อวันที่ 6 กันยายน พ.ศ. 2523 ที่จังหวัดเชียงใหม่ สำเร็จการศึกษาระดับปริญญาบัณฑิต หลักสูตรวิศวกรรมศาสตรบัณฑิต (วศ.บ.) สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2545 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2549 ปัจจุบันทำงานอยู่ที่ บริษัท รอยเตอร์ซอฟต์แวร์ (ประเทศไทย) จำกัด

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย