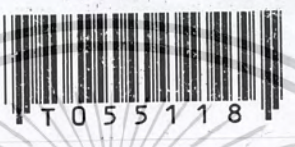
โปรแกรมรีเวิร์สเว็บพร็อกซี่แบบปลอดภัย

SECURE REVERSE WEB PROXY



นาย เขมรินทร์ คงประเสริฐ

นาย ไววิชา บุราคร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

โปรแกรมรีเวิร์สเว็บพร็อกซี่แบบปลอดภัย

SECURE REVERSE WEB PROXY

โดย

นาย เขมรินทร์ คงประเสริฐ

นาย ไววิชา บุราคร

อาจารย์ที่ปรึกษา

อาจารย์ ธนา หงษ์สุวรรณ

อาจารย์ อัครเดช วัชระภูพงษ์

อาจารย์ ธนัญชัย ตรีภาค

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมรีเวิร์สเว็บพร็อกซี่แบบปลอดภัย

SECURE REVERSE WEB PROXY

ผู้จัดทำ

1. นาย เขมรินทร์ คงประเสริฐ รหัสประจำตัว 43010041
2. นาย ไววิชา บุราคร รหัสประจำตัว 43010417

_____อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

_____อาจารย์ที่ปรึกษา

(อาจารย์ อัครเดช วัชระภูพงษ์)

_____อาจารย์ที่ปรึกษา

(อาจารย์ ธนัญชัย ตรีภาค)

โปรแกรมรีเวิร์สเว็บพร็อกซี่แบบปลอดภัย

นายเขมรินทร์ คงประเสริฐ	43010041
นายไววิชา บุราคร	43010417
อาจารย์ ธนา หงษ์สุวรรณ	อาจารย์ที่ปรึกษา
อาจารย์ อัครเดช วัชระภูพงษ์	อาจารย์ที่ปรึกษา
อาจารย์ ธนัญชัย ตรีภาค	อาจารย์ที่ปรึกษา
ปีการศึกษา 2546	

บทคัดย่อ

เว็บเซิร์ฟเวอร์มักเป็นเป้าหมายในการโจมตีผ่านระบบเครือข่าย ซึ่งการป้องกันการโจมตีในขั้นต้นสามารถกระทำได้คือการติดตั้งไฟร์วอลล์ระหว่างเครือข่ายภายนอกและเว็บเซิร์ฟเวอร์ แต่การป้องกันการโจมตีโดยใช้ไฟร์วอลล์แบบการกรองแพ็กเก็ต (Packet Filtering) ยังไม่สามารถทำได้ ทั้งนี้เป็นเพราะการโจมตีผ่านโปรโตคอลเอชทีทีพี (HTTP) มีรูปแบบที่ไม่ตายตัวซึ่งขึ้นอยู่กับระบบปฏิบัติการ, รุ่น และชนิดของเว็บเซิร์ฟเวอร์นั้น ๆ

รีเวิร์สเว็บพร็อกซี่ทำหน้าที่เป็นตัวกลางคอยส่งผ่านข้อมูลระหว่างเซิร์ฟเวอร์และไคลเอ็นต์ มีความสามารถในการแคช (cache) ข้อมูลจากเว็บเซิร์ฟเวอร์และการทำโหลดบาลานซ์ (Load Balance) ในกรณีที่มีเซิร์ฟเวอร์หลายตัวได้อีกด้วย ผู้วิจัยจึงนำคุณสมบัติของการเป็นตัวกลางส่งผ่านข้อมูล นำข้อมูลมาตรวจสอบโดยกรรมวิธีของ Boyer-Moore ว่าเข้าข่ายการบุกรุกและควรนำส่งต่อให้แก่เว็บเซิร์ฟเวอร์หรือไม่ อันจะทำให้ระบบโดยรวมมีความปลอดภัยมากยิ่งขึ้น โดยเรียกโปรแกรมที่พัฒนาขึ้นมาว่า “โปรแกรมรีเวิร์สเว็บพร็อกซี่แบบปลอดภัย”

SECURE REVERSE WEB PROXY

Mr. Khemarin Khongprasert
Mr. Waiwicha Burakorn
Mr. Thana Hongsuwan Advisor
Mr. Akkradach Watcharapupong Advisor
Mr. Thananchai Treepak Advisor
Academic Year 2003

ABSTRACT

The most intrusion on network is web server intrusion. The solution to protect them is firewalls. But the firewalls (packet filtering firewall) can not protect web server from intrusion via HTTP protocol because there are many various types of HTTP intrusion.It depends on operating system and web server.

Among client and web server is Reverse Web Proxy which has an ability- redirect , cache and load balance. This project focus on data redirection using Boyer Moore algorithm to detect intrusion , That can improve overall of security system's performance. I has called this program "Secure Reverse Web Proxy".

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้มิอาจเสร็จสมบูรณ์ได้ หากไม่ได้รับคำแนะนำ ความช่วยเหลือ จากหลายๆ ฝ่ายด้วยกัน โดยเฉพาะ ท่านอาจารย์ที่ปรึกษาทั้งสามท่าน คือ อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ อัครเดช วัชระภูพงษ์ และ อาจารย์ธนัญชัย ตรีภาค คณะผู้จัดทำจักขอบพระคุณยิ่งสำหรับทุกสิ่งทุกอย่าง

ขอขอบคุณภาควิศวกรรมคอมพิวเตอร์ ที่ได้เอื้อเฟื้อสถานที่ และขอขอบคุณพี่ๆ และ เพื่อนๆ ทุกคนที่ให้คำปรึกษา และเป็นกำลังใจให้กันตลอดมา

ต้องขอขอบพระคุณ คุณบิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้ อบรม สั่งสอน เลี้ยงดู มาเป็นอย่างดี พร้อมทั้งให้โอกาสการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา นับเป็นพระคุณที่หาที่เปรียบไม่ได้ ทางคณะผู้จัดทำขอกราบขอบพระคุณมา ณ ที่นี้

เขมรินทร์ คงประเสริฐ
ไววิชา บุราคร

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เว็บเซิร์ฟเวอร์ (Web Server) มักเป็นเป้าหมายในการโจมตีผ่านระบบเครือข่าย ซึ่งวิธีการป้องกันการโจมตีสามารถกระทำได้คือ ติดตั้งไฟร์วอลล์ (Firewall) แต่การป้องกันโดยวิธีนี้ยังไม่สามารถป้องกันการโจมตีผ่านระบบเว็บซึ่งใช้โปรโตคอล HTTP เป็นหลัก และมีรูปแบบที่อยู่เหนือระดับความสามารถของไฟร์วอลล์ที่จะป้องกันได้ เพราะมีรูปแบบที่ไม่ตายตัวตามระบบปฏิบัติการและรุ่นของซอฟต์แวร์ที่ทำเป็นเว็บเซิร์ฟเวอร์ (Server) ผู้วิจัยจึงพัฒนา รีเวิร์สเว็บพร็อกซี่ซึ่งมีความสามารถในการตรวจจับการบุกรุกและการโจมตีเว็บเซิร์ฟเวอร์ ซึ่งเป็นระดับแอพพลิเคชั่น (Application) จะทำให้ระบบเว็บเซิร์ฟเวอร์มีความปลอดภัยมากยิ่งขึ้น

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อศึกษาแนวคิดและการประยุกต์ของรีเวิร์สเว็บพร็อกซี่
1.2.2 เพื่อศึกษาแนวทางการป้องกันการโจมตีเว็บเซิร์ฟเวอร์
1.2.3 เพื่อพัฒนาโปรแกรมรีเวิร์สเว็บพร็อกซี่ต้นแบบที่ช่วยป้องกันการโจมตีเว็บเซิร์ฟเวอร์

1.3 ขอบเขตและข้อจำกัดของงานวิจัย

งานวิจัยนี้เป็นการพัฒนาโปรแกรมรีเวิร์สเว็บพร็อกซี่ที่จะทำงานกับเว็บเซิร์ฟเวอร์ ในการทดลองนี้จะใช้เว็บเซิร์ฟเวอร์อยู่สองชนิดคือ Apache และ IIS แต่จะทำงานได้กับเว็บเซิร์ฟเวอร์ครั้งละหนึ่งเครื่องเท่านั้น โดยจะใช้งานโปรโตคอลเอชทีทีพี (HTTP Protocol) เป็นหลัก การทำงานจะใช้หลักการของรีเวิร์สพร็อกซี่แต่จะเน้นในส่วนของการเพิ่มความปลอดภัยเป็นหลักโดยไม่คำนึงถึงการทำแคชชิ่ง (caching) และไม่รองรับการทำโหลดบานลานซ์ (load balance)

การเพิ่มความปลอดภัยนี้จะต้องอ่านข้อมูลในการร้องขอ (request) ที่เข้ามายังเว็บเซิร์ฟเวอร์ ดังนั้นหากข้อมูลนั้นเข้ารหัสมา เช่น เว็บเซิร์ฟเวอร์มีการใช้งานเอสเอสแอล (SSL) ก็จะไม่สามารถตรวจสอบการร้องขอนั้นได้ การรองรับการโจมตีที่มีนั้นจะรองรับเฉพาะที่มีข้อมูลอยู่แล้วเท่านั้นในที่นี้คือเฉพาะที่มีอยู่ในรูล ซึ่งสามารถปรับแต่ง เพิ่ม ลด อัพเดท ได้ในภายหลัง และ จะไม่สามารถรองรับการโจมตีประเภทที่เป็นการโจมตีเพื่อปิดบริการ (denial of service : DOS)
ในส่วนของการทำเชื่อมต่อระหว่างเครื่องที่ให้บริการกับเครื่องรับบริการนั้นจะต้องไม่ทำ การเชื่อมต่อแบบยังคงอยู่ (persistence connection)

1.4 เครื่องมือที่ใช้ในการพัฒนา

1. GNU compiler
2. VI editor
3. Apache web server
4. IIS web server
5. Vmware
6. Ethereal
7. Stealth

1.5 วิธีการดำเนินงาน

การดำเนินงานเริ่มด้วยการศึกษาทฤษฎีและแนวความคิดต่างๆ ซึ่งเกี่ยวกับรีเวิร์สพร็อกซี่และการโจมตีและการป้องกันที่จะสร้างขึ้น ส่วนของรายละเอียดต่าง ๆ ในบทที่ 2 จากนั้นจะนำความรู้ที่ได้ศึกษามาข้างต้นมาวิเคราะห์ และ ออกแบบเพื่อสร้างโปรแกรม ซึ่งรายละเอียดของโครงสร้างโปรแกรมนั้นจะอยู่ในบทที่ 3

หลังจากนั้นจะเริ่มเข้าสู่ขั้นตอนการทดลองและค้นหาข้อผิดพลาดของโปรแกรมที่ได้ออกแบบมาเพื่อนำมาแก้ไขและพัฒนาต่อไป

เมื่อได้ทำการทดลองและแก้ไขข้อผิดพลาดต่าง ๆ เรียบร้อยแล้วก็จะนำมาสรุปผลการดำเนินงานของงานวิจัยชิ้นนี้ว่ามีข้อดี ข้อเสียอย่างไร รวมทั้งแนวทางการปรับปรุงและพัฒนาในอนาคต พร้อมทั้งจะกล่าวสิ่งที่ได้รับจากงานวิจัย ซึ่งจะอธิบายรายละเอียดในบทที่ 4 และ 5

บทที่ 2
ทฤษฎีและหลักการ

2.1 บทนำ

ในเกือบจะทุกองค์กรจะมีเว็บเซิร์ฟเวอร์เพื่อใช้แสดงข้อมูลและรายละเอียดต่าง ๆ องค์กรนั้น ดังนั้นเว็บของแต่ละองค์กรจึงหมายถึงหน้าตาขององค์กรนั้น ๆ ด้วย กรณีที่มีข่าวว่าเว็บเซิร์ฟเวอร์ขององค์กรนั้นถูกบุกรุก จะทำให้ความน่าเชื่อถือของข้อมูลจากเว็บขององค์กรเหล่านั้นลดลงไปอย่างมาก การป้องกันการโจมตีสามารถกระทำได้โดยการติดตั้งไฟร์วอลล์ระหว่างเครือข่ายภายนอก และ เว็บ-เซิร์ฟเวอร์ แต่การป้องกันการโจมตีโดยใช้ไฟร์วอลล์แบบการกรองแพ็กเก็ต (Packet Filtering) และ สเตทฟูล(Stateful) ยังไม่สามารถทำได้ ทั้งนี้เป็นเพราะการโจมตีผ่านโปรโตคอลเอชทีทีพี (HTTP) เป็นการโจมตีในระดับแอพพลิเคชั่น (Application Layer) และมีรูปแบบที่ไม่ตายตัวซึ่งขึ้นอยู่กับระบบปฏิบัติการ (Linux, Windows, Solaris และอื่น ๆ) ชนิดของเว็บเซิร์ฟเวอร์ (Apache , IIS , Atari800 , Samba และอื่นๆ) และรุ่นของเว็บเซิร์ฟเวอร์ด้วย

แนวทางในการป้องกันการโจมตีสามารถทำได้ดังนี้คือ เพิ่มความสามารถให้ไฟร์วอลล์ ให้สามารถตรวจดูข้อมูลในระดับแอพพลิเคชั่น (Appication Firewall) ซึ่งไฟร์วอลล์ประเภทนี้จะมีราคาค่อนข้างสูง หรือ เพิ่มส่วนการตรวจจับการบุกรุกให้แก่เว็บเซิร์ฟเวอร์ซึ่งจะเป็นเพิ่มภาระให้แก่ตัวเว็บเซิร์ฟเวอร์เป็นอย่างมาก แนวคิดและวิธีการทำงานของรีเวิร์สเว็บพร็อกซี (Reverse Web Proxy) จึงถูกนำมาประยุกต์ใช้ในโครงงานนี้

รีเวิร์สเว็บพร็อกซีทำหน้าที่เป็นตัวกลางคอยส่งผ่านข้อมูลระหว่างเซิร์ฟเวอร์และไคลเอ็นต์ มีความสามารถในการแคช (cache) ข้อมูลจากเว็บเซิร์ฟเวอร์และการทำโหลดบาลานซ์ (Load Balance) ในกรณีที่มีเซิร์ฟเวอร์หลายตัวได้อีกด้วย ผู้วิจัยจึงนำคุณสมบัติของการเป็นตัวกลางส่งผ่านข้อมูล นำข้อมูลมาตรวจสอบโดยใช้ Boyer-Moore Algorithms ซึ่งเป็นการทำ pattern matching ที่มีความรวดเร็ว ตรวจสอบว่าเข้าข่ายการบุกรุกและควรนำส่งต่อให้แก่เว็บเซิร์ฟเวอร์หรือไม่ โดยถ้าเป็นการบุกรุกก็จะตอบกลับไปยังเครื่องผู้บุกรุกว่า HTTP 404 Not Found ซึ่งหมายถึงไม่พบ URL ที่ต้องการ จะทำให้ผู้บุกรุกสับสนได้ว่าการโจมตีนั้นประสบความสำเร็จหรือไม่ แต่ถ้าไม่เข้าข่ายการบุกรุกก็จะส่ง request ของ ไคลเอ็นต์ไปให้แก่เว็บเซิร์ฟเวอร์ตามปกติ

2.2 เว็บพร็อกซี่ (Web Proxy)

การใช้งานหลัก ๆ ของพร็อกซี่ประเภทนี้คือ การอนุญาตให้เครื่องไคลเอ็นต์ภายในเครือข่ายและหลังไฟร์วอลล์ใช้งานอินเตอร์เน็ตผ่านตัวมันได้ โดยเครื่องพร็อกซี่จะรอรับการร้องขอจากเครื่องไคลเอ็นต์ที่ อยู่หลังไฟร์วอลล์และจะส่งต่อการร้องขอนี้ไปยังเครื่องเซิร์ฟเวอร์ปลายทางด้านนอกไฟร์วอลล์ จากนั้นก็รอรับการตอบกลับมาแล้วส่งไปให้แก่เครื่องไคลเอ็นต์ต่อไป

ปกติแล้วเครื่องไคลเอ็นต์ทุกเครื่องที่อยู่ในซับเน็ตเดียวกันจะใช้งานเครื่องพร็อกซี่ตัวเดียวกัน ซึ่งจะทำให้เครื่องพร็อกซี่สามารถแคชข้อมูลต่าง ๆ ได้อย่างมีประสิทธิภาพซึ่งจะเพิ่มขึ้นตามจำนวนของเครื่องไคลเอ็นต์ คนที่ใช้งานเครื่องพร็อกซี่จะรู้สึกราวกับว่าเค้าได้รับการตอบรับโดยตรงจากเครื่องเซิร์ฟเวอร์ที่ร้องขอไปแต่ว่ามีความรวดเร็วมากยิ่งขึ้น

เครื่องไคลเอ็นต์ที่ใช้งานไพรเวตไอพีก็สามารถที่จะใช้งานอินเตอร์เน็ตได้เช่นกันเพียงแค่รู้ไอพีแอดเดรสของเครื่องพร็อกซี่เท่านั้น ในองค์กรที่ใช้ไอพีแอดเดรสในช่วงที่เป็นไพรเวตไอพีเช่นคลาส A ซึ่งเป็นลักษณะ 10.*.*.* ก็สามารถที่จะใช้งานอินเตอร์เน็ตได้ถ้าเครื่องพร็อกซี่นั้นสามารถมองเห็นได้จากทั้งฝั่งที่เป็นไพรเวตไอพีภายในองค์กรและฝั่งอินเตอร์เน็ต

เครื่องพร็อกซี่ส่วนใหญ่จะถูกติดตั้งมาให้รองรับงานบริการเดียวเป็นหลัก ซึ่งพร็อกซี่เซิร์ฟเวอร์นั้นสามารถอนุญาตหรือจะปฏิเสธการร้องขอที่ถูกต้องตามที่กำหนดไว้ได้ตัวอย่างเช่นเครื่องพร็อกอนุญาตให้การร้องขอไปยัง FTP เซิร์ฟเวอร์ผ่านไปได้ในขณะที่ไม่อนุญาตให้ร้องขอไปยัง HTTP เซิร์ฟเวอร์

การใช้งานเว็บพร็อกซี่

เราสามารถใช้งานพร็อกซี่เซิร์ฟเวอร์ได้หลาย ๆ อย่างโดยรวมทั้งความสามารถดังนี้

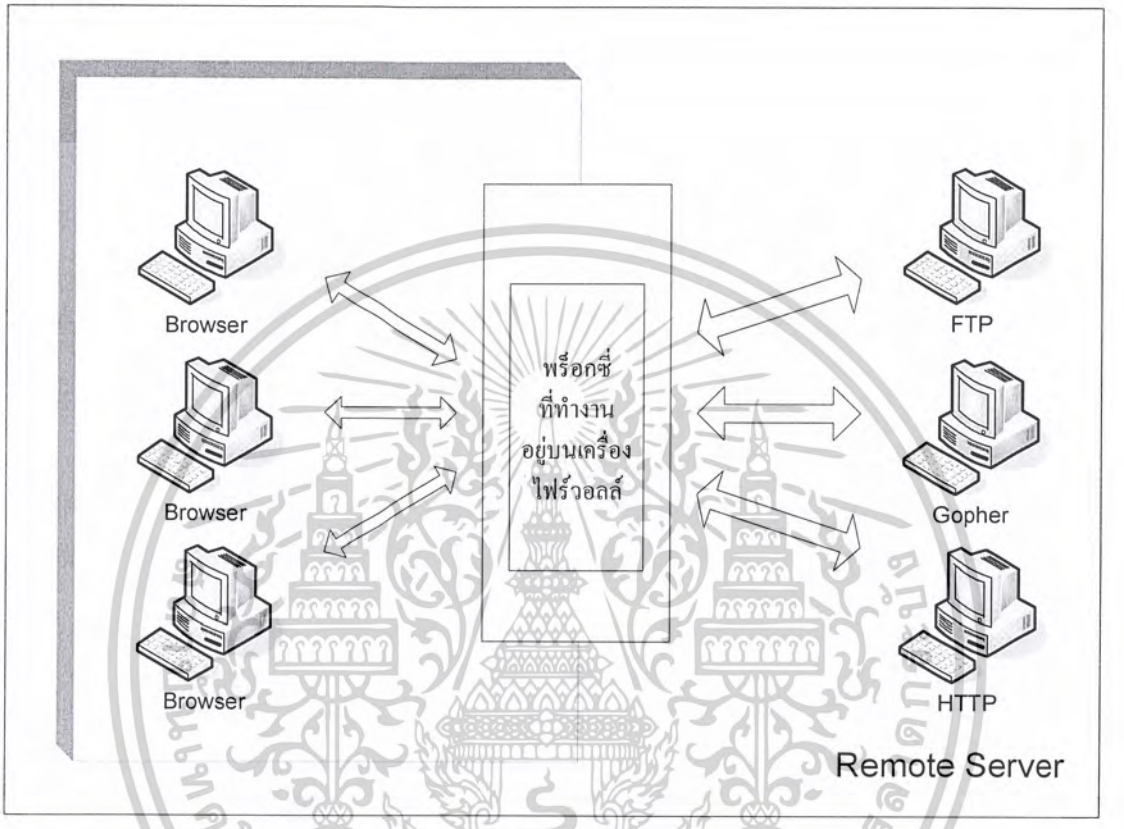
1. อนุญาตและจำกัดการใช้งานอินเตอร์เน็ตของเครื่องไคลเอ็นต์โดยวิธีการตรวจสอบหมายเลขไอพี
2. แคชข้อมูลจากภายนอกให้เครื่องภายในที่ต้องการข้อมูลเหมือน ๆ กันเรียกใช้ได้เร็วยิ่งขึ้น
3. สามารถควบคุมให้การเข้าใช้งานอินเตอร์เน็ตและซับเน็ตที่ต้องการได้โดยกำหนด URL เป็นหลัก
4. ให้บริการการใช้งานอินเตอร์เน็ตแก่บริษัทที่ใช้ไอพีในช่วงไพรเวตไอพี
5. แปลงข้อมูลให้อยู่ในรูปแบบของ HTML ซึ่งสามารถอ่านได้โดยบราวเซอร์

การเรียกใช้อินเตอร์เน็ตของบราวเซอร์

มีเครื่องบางเครื่องในเครือข่ายของเราที่ไม่สามารถเข้าใช้งานอินเตอร์เน็ตได้โดยตรงตัวอย่างเช่นบราวเซอร์บางตัวที่ไม่สามารถเข้าใช้งานอินเตอร์เน็ตโดยตรงเป็นเพราะว่าเครื่องที่ใช้งานอยู่นั้นอยู่ภายใต้การปกป้องโดยไฟร์วอลล์ซึ่งในกรณีนี้พร็อกซี่เซิร์ฟเวอร์สามารถที่จะไปดึงข้อมูลที่ต้องการมาให้แก่บราวเซอร์ได้

จากรูป 2.1 พร็อกซี่เซิร์ฟเวอร์ทำงานอยู่บนเครื่องไฟร์วอลล์และเชื่อมต่อไปยังอินเตอร์เน็ตโดยใช้ซอฟต์แวร์ของไฟร์วอลล์ ซึ่งเราสามารถที่จะตั้งเครื่องพร็อกซี่ไว้ที่เครื่องอื่นภายในเครือข่ายที่สามารถเรียกใช้อินเตอร์เน็ตโดยตรงได้ หรือจะติดตั้งไว้ยังเครื่องที่อยู่ในไฟร์วอลล์ก็ได้

เมื่อเครื่องพร็อกซี่นั้นรับการร้องขอจากบราวเซอร์ในรูปแบบของ URL เครื่องพร็อกซี่ก็จะไปดึงข้อมูลจากตำแหน่งที่ให้ไว้ใน URL นั้นและแปลงเป็นรูปแบบของ HTML แล้วจึงส่งกลับมายังบราวเซอร์ที่อยู่ด้านหลังไฟร์วอลล์



รูปที่ 2-1 พร็อกซี่ที่ทำงานอยู่บนเครื่องไฟร์วอลล์

การแคชข้อมูล

โดยปกติแล้วเครื่องไคล์เอ็นต์ภายในซับเน็ตเดียวกันจะใช้พร็อกซี่เซิร์ฟเวอร์ตัวเดียวกัน ซึ่งเครื่องพร็อกซี่บางตัวจะแคชข้อมูลสำหรับเครื่องไคลเอ็นต์ที่อยู่เครือข่ายเดียวกัน การแคชข้อมูลหมายถึงการเก็บสำเนาข้อมูลจากอินเตอร์เน็ตมาไว้ยังเครื่องภายใน ดังนั้นเครื่องพร็อกซี่จึงไม่จำเป็นที่จะต้องไปร้องขอข้อมูลเหล่านี้มาซ้ำแล้วซ้ำเล่า เพียงแต่นำส่งข้อมูลเหล่านี้ไปให้เครื่องไคลเอ็นต์ที่ต้องการข้อมูลอันเดียวกันเท่านั้น

การแคชนั้นจะเป็นประโยชน์แก่พร็อกซี่เซิร์ฟเวอร์เองมากกว่าตัวไคล์เอ็นต์ เนื่องจากทำให้ประหยัดพื้นที่ในการจัดเก็บ เพราะว่าจะจัดเก็บสำเนาเพียงชุดเดียว การแคชข้อมูลเก็บไว้จะมีความหมายถ้าข้อมูลที่แคชเอาไว้ถูกเรียกใช้จากเครื่องไคลเอ็นต์หลาย ๆ เครื่องอันจะทำให้เกิดประโยชน์มาก ผู้ดูแลระบบสามารถที่จะคาดเดาได้ว่าข้อมูลชนิดใดบ้างที่เป็นหรือไม่เป็นประโยชน์ถ้ามีการแคชไว้เป็นเวลานาน

การแคชนั้นสามารถทำให้การเรียกเว็บในขณะที่เครือข่ายอินเตอร์เน็ตไม่สามารถใช้งานได้ตราบเท่าที่เครื่องที่ร้องขอสามารถเชื่อมต่อกับเครื่องพร็อกซี่ได้เช่นกรณีที่เครื่องเว็บเซิร์ฟเวอร์ไม่สามารถให้บริการได้แต่เราก็สามารถที่จะดึงข้อมูลบางส่วนมาจากแคชได้

การควบคุมการใช้งานอินเตอร์เน็ต

เมื่อใช้พร็อกซี่นั้นเราสามารถที่จะกรองการใช้งานระดับโปรโตคอลของเครื่องไคลเอนต์ได้ พร็อกซี่สามารถที่จะควบคุมการใช้บริการในแต่ละโฮสต์และโดเมนได้เช่น

- ตัดสินใจว่าการร้องขอรูปแบบใดบ้างที่ควรจะให้ใช้งาน
- กำหนด URL ที่จะใช้ในการกรองหรืออนุญาตให้เรียกใช้ผ่านเครื่องเซิร์ฟเวอร์ได้
- กำหนดว่าโปรโตคอลใดที่เครื่องไคล์เอ็นต์สามารถเรียกใช้งานได้โดยขึ้นอยู่กับหมายเลขไอพีของเครื่องไคลเอ็นต์นั้น ๆ

การปรับแต่งบราวเซอร์ให้ใช้งานพร็อกซี่

บราวเซอร์ที่ต้องการใช้งานพร็อกซี่นั้นจะต้องบอกให้บราวเซอร์นั้นส่งการร้องขอผ่านไปยังเครื่องพร็อกซี่ บราวเซอร์ส่วนใหญ่ยอมให้เราตั้งค่านี้และมันจะส่งการร้องขอผ่านไปยังพร็อกซี่เซิร์ฟเวอร์ ในแต่ละบราวเซอร์นั้นก็จะมีวิธีการกำหนดค่าได้หลายรูปแบบทั้งแบบโดเมนเนมและหมายเลขไอพี อย่างไรก็ตามถ้าไม่มีการตั้งค่าให้แก่บราวเซอร์ตัวบราวเซอร์ก็จะไม่ส่งค่าไป

การทำงานของบราวเซอร์ที่เชื่อมต่อกับเซิร์ฟเวอร์ตามปกติ

เมื่อเครื่องแต่ละเครื่องมีหมายเลขไอพีเป็นของตัวเองและเชื่อมต่อกับเซิร์ฟเวอร์โดยตรงผ่านอินเตอร์เน็ต เมื่อบราวเซอร์ได้ร้องขอแบบ HTTP ตัวเซิร์ฟเวอร์ได้รับ path และ keyword ที่ใช้ร้องขอ ซึ่งตัว path จะอ้างถึงเอกสารหรือโปรแกรมพวก CGI หรือทรัพยากรอื่น ๆ

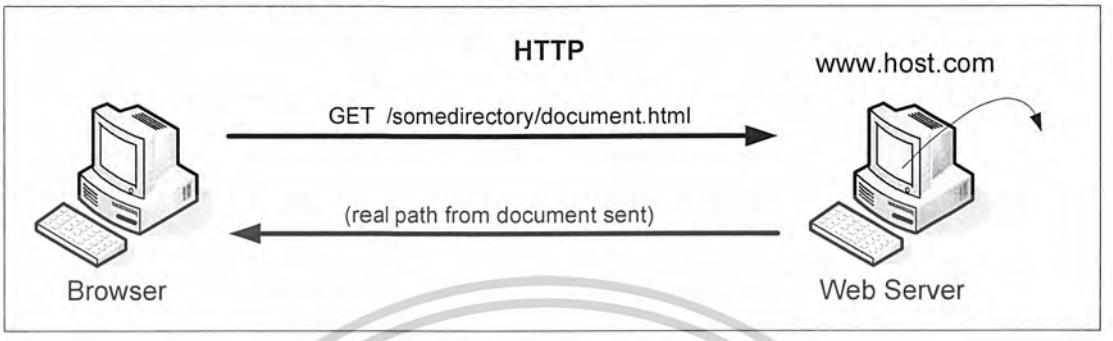
เมื่อผู้ใช้ป้อน :

http://www.host.com/path/doc.html

บราวเซอร์จะเปลี่ยนเป็น :

GET /path/doc.html

บราวเซอร์จะเชื่อมต่อกับเซิร์ฟเวอร์จากนั้นจะส่งคำสั่งเข้าไปแล้วรอการตอบรับ ในตัวอย่างนี้บราวเซอร์ส่งการร้องขอไปยัง HTTP เซิร์ฟเวอร์และบอกว่าต้องการทรัพยากรตัวไหนในเซิร์ฟเวอร์นั้น โดยไม่มีการบอกถึงโปรโตคอลหรือชื่อโฮสต์ใน URL จากนั้นเซิร์ฟเวอร์จะตอบกลับมาเป็นเอกสารที่ต้องการหรืออาจจะส่งข้อความว่าเป็นการร้องขอที่ผิดพลาดได้



รูปที่ 2-2 การเรียกใช้งานเว็บตามปกติของบราวเซอร์

การสื่อสารโดยผ่านพร็อกซี่เซิร์ฟเวอร์

พร็อกซี่นั้นจะทำตัวเป็นทั้งเซิร์ฟเวอร์และไคลเอ็นต์โดยจะทำตัวเป็นเซิร์ฟเวอร์เมื่อรับการร้องขอ HTTP จากบราวเซอร์และเป็นไคลเอ็นต์เมื่อส่งการร้องขอไปยังเซิร์ฟเวอร์ภายนอกเพื่อดึงข้อมูลเหล่านี้มา พร็อกซี่นั้นจะทำข้อมูลในส่วน header ที่ได้รับมาส่งผ่านไปให้ยังเซิร์ฟเวอร์โดยไม่มีการแก้ไขใด ๆ ซึ่งหมายความว่า บราวเซอร์นั้นได้ส่งข้อมูลไปให้เซิร์ฟเวอร์ได้อย่างครบถ้วนเมื่อผ่านพร็อกซี่

เมื่อบราวเซอร์ติดต่อผ่านพร็อกซี่บราวเซอร์จะใช้ HTTP โปรโตคอลในการติดต่อกับพร็อกซี่ ถึงแม้ว่าผู้ใช้งานต้องการจะติดต่อไปยังเซิร์ฟเวอร์โดยใช้โปรโตคอลอื่นก็ตาม เช่น FTP เป็นต้น

แทนที่บราวเซอร์จะส่งเฉพาะ path และ keyword ที่ต้องการไปยังพร็อกซี่เซิร์ฟเวอร์ บราวเซอร์จะส่ง URL แบบเต็มไปยังพร็อกซี่หลังจากนั้นพร็อกซี่จะแปลงการร้องขอนั้นให้อยู่ในรูปแบบที่ใช้ในการติดต่อกับตัวเซิร์ฟเวอร์

ผู้ใช้ป้อน :

http://www.host.com/path/doc.html

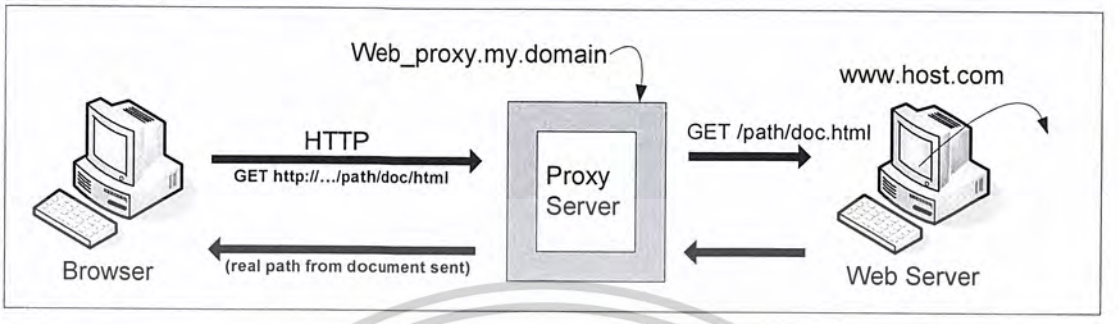
บราวเซอร์ส่งไปให้พร็อกซี่ :

GET http://www.host.com/path/doc.html

พร็อกซี่ส่งไปยังเซิร์ฟเวอร์ :

GET /path/doc.html

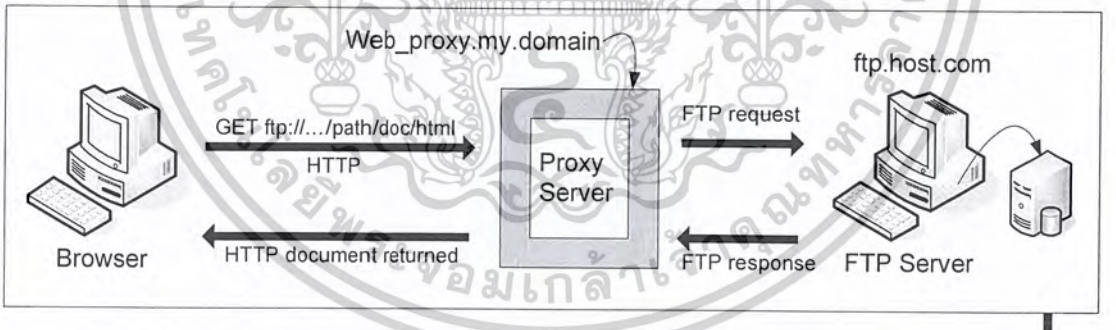
พร็อกซี่จะเชื่อมต่อกับเซิร์ฟเวอร์จากนั้นจะส่งคำสั่งเข้าไปแล้วรอการตอบรับ ในตัวอย่างนี้พร็อกซี่ส่งการร้องขอไปยัง HTTP เซิร์ฟเวอร์และบอกว่าต้องการทรัพยากรตัวไหนในเซิร์ฟเวอร์นั้น จากนั้นเซิร์ฟเวอร์จะตอบกลับมาเป็นเอกสารที่ต้องการแล้วตัวพร็อกซี่ก็ส่งข้อมูลที่ตอบกลับมานี้ไปให้แก่บราวเซอร์อีกที ดังที่แสดงอยู่ในรูป 2.3



รูปที่ 2-3 การร้องขอไปยัง HTTP เซิร์ฟเวอร์ผ่านพร็อกซี่

บราวเซอร์ร้องขอ FTP โดยผ่านพร็อกซี่

รูปที่ 2-4 แสดงให้เห็นถึงการร้องขอผ่านพร็อกซี่โดยใช้ HTTP ถึงแม้ว่าข้อมูลที่ร้องขอจะอยู่บน FTP เซิร์ฟเวอร์ พร็อกซี่สามารถดูได้จาก URL ว่าควรจะทำการเชื่อมต่อแบบ FTP เมื่อพร็อกซี่สร้างการเชื่อมต่อและดึงไฟล์มาจาก FTP เซิร์ฟเวอร์ก็จะส่งไฟล์กลับมายังบราวเซอร์โดยใช้ HTTP หรือในอีกกรณีหนึ่งคือ พร็อกซี่จะส่งค่าเป็นลิสต์ของไดเรกทอรีกลับมาเป็นรูปแบบของเอกสาร HTML



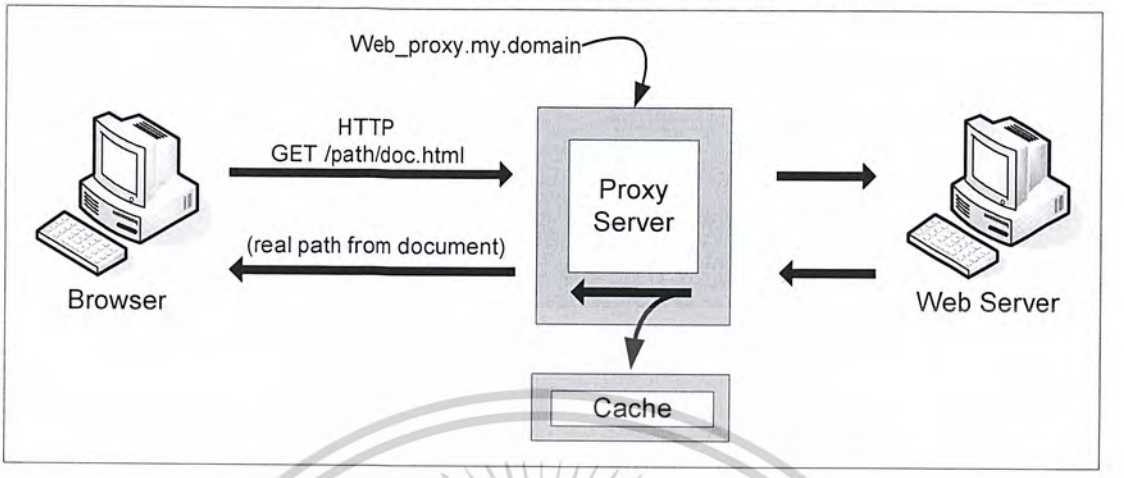
รูปที่ 2-4 การร้องขอจาก FTP เซิร์ฟเวอร์ผ่านพร็อกซี่

ข้อดีในการแคชข้อมูล

เมื่อบราวเซอร์ร้องขอไฟล์พร็อกซี่จะตรวจสอบที่แคชของตัวเองว่ามีข้อมูลนี้อยู่หรือไม่ ถ้ามีไฟล์อยู่ในแคชพร็อกซี่ก็จะส่งไฟล์นี้ไปยังบราวเซอร์ถ้าเราต้องการที่จะแคชข้อมูลเราต้องตัดสินใจว่า

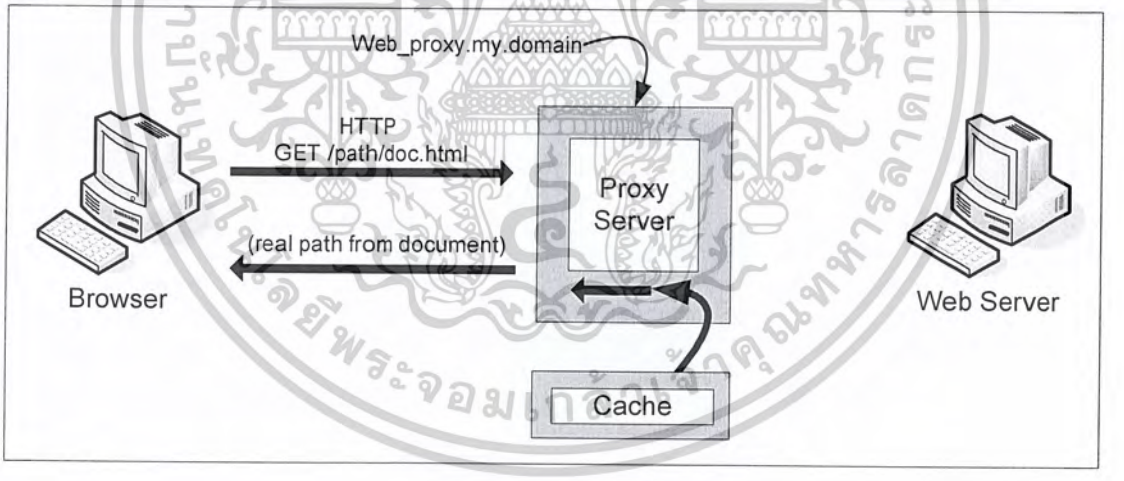
- ข้อมูลอะไรบ้างที่ถูกเรียกใช้บ่อยเพียงพอที่จะเก็บเอาไว้ในแคช
- เราควรจะเก็บข้อมูลไว้นานแค่ไหนก่อนที่จะไปดึงข้อมูลที่ใหม่กว่านี้มาเก็บไว้

รูปที่ 2-5 จะแสดงให้เห็นวิธีที่พร็อกซี่แคชข้อมูลที่ดึงมาจากเซิร์ฟเวอร์แล้วเก็บเอาไว้ ซึ่งเครื่องไคลเอ็นต์สามารถร้องขอและดึงข้อมูลไปจากแคชได้ถ้ามีการร้องขอข้อมูลอันเดิมในครั้งหลัง



รูปที่ 2-5 การแคชข้อมูลของพร็อกซี่

ถ้าพบข้อมูลในแคชว่าเป็นข้อมูลที่ update แล้วพร็อกซี่ก็ไม่จำเป็นต้องเชื่อมต่อไปยังเซิร์ฟเวอร์เพื่อร้องขอข้อมูล



รูปที่ 2-6 การนำข้อมูลจากแคชส่งให้บราวเซอร์

การแคชข้อมูลสามารถช่วยลดเวลารอในการรอคอยของผู้ใช้งานเมื่อเขาร้องขอข้อมูลที่อยู่บนอินเตอร์เน็ต พร็อกซี่สามารถที่จะให้บริการข้อมูลเหล่านี้ได้รวดเร็วกว่าเซิร์ฟเวอร์ที่อยู่ห่างออกไป และเมื่อมีการแคชข้อมูลที่ผู้ใช้ส่วนใหญ่ที่ต้องการก็จะสามารถช่วยลดการเชื่อมต่อและการใช้งานเครือข่ายภายนอกได้อีกด้วย

2.3 รีเวิร์สพร็อกซี่ (Reverse Proxy)

รีเวิร์สพร็อกซี่คืออะไร

รีเวิร์สพร็อกซี่เป็นพร็อกซี่ที่ทำงานบนฝั่งเซิร์ฟเวอร์แทนที่จะทำงานบนฝั่งไคลเอ็นต์จึงเรียกว่าเป็น การทำงานแบบรีเวิร์ส รีเวิร์สพร็อกซี่เป็นแอพพลิเคชั่นพร็อกซี่สำหรับเซิร์ฟเวอร์ที่ใช้โปรโตคอล HTTP โดยทำหน้าที่เป็นเกตเวย์ของเว็บเซิร์ฟเวอร์หรือ เว็บเซิร์ฟเวอร์ฟาร์ม โดยทำหน้าที่เป็นไอพีแอดเดรสที่ใช้สำหรับรับการร้องขอจากภายนอก ไฟร์วอลล์ทำงานร่วมกับรีเวิร์สพร็อกซี่เพื่อช่วยให้แน่ใจว่ารีเวิร์สพร็อกซี่จะสามารถเข้าถึง เว็บเซิร์ฟเวอร์ที่ถูกซ่อนอยู่ด้านหลังได้ โดยมุมมองจากภายนอกแล้วไคลเอ็นต์จะเห็นรีเวิร์สพร็อกซี่เป็นเว็บเซิร์ฟเวอร์เครื่องเท่านั้น

ข้อดีของรีเวิร์สพร็อกซี่

ประโยชน์ที่เห็นได้ชัดที่สุดจากการใช้รีเวิร์สพร็อกซี่คือ ทำให้ไคลเอ็นต์สามารถเข้าใช้งานเซิร์ฟเวอร์ด้วยวิธี single point access ทำให้สามารถเพิ่มเลเยอร์ ต่าง ๆ เพื่อช่วยในการป้องกันการโจมตีเซิร์ฟเวอร์ได้

การเชื่อมต่อแบบ single point ทำให้สามารถกำหนดความสามารถในการเข้าถึงเซิร์ฟเวอร์ต่าง ๆ ของผู้ใช้แต่ละคนได้ ด้วยการกำหนดเพียงครั้งเดียว

ในการเปลี่ยนเซิร์ฟเวอร์หรือ ชื่อโฮสต์สามารถทำได้ง่าย เพราะการเปลี่ยนแปลงที่เกิดขึ้นไม่ส่งผลกระทบต่อไคลเอ็นต์ เนื่องจากการเปลี่ยนแปลงเหล่านี้เกิดขึ้นภายในรีเวิร์สพร็อกซี่ทำได้โดยเปลี่ยน rules หรือ mapping ดังนั้นจึงไม่ต้องเสียเวลาในการเปลี่ยนชื่อเซิร์ฟเวอร์ที่ DNS

แนวความคิดในการเรียกใช้งานแบบ single point ช่วยให้สามารถทำ load balancing และจัดการกับ failover ได้ด้วยการใช้ DNS round robin schema หรือใช้ hardware หรือ software เช่น F5 Network Big IP, Cisco's Content Switch หรือ Macromedia's Cluster Cat เป็นต้น

การใช้รีเวิร์สพร็อกซี่ทำให้สามารถเรียกใช้งาน แอพพลิเคชั่นต่าง ๆ ที่อยู่บนหลายระบบปฏิบัติการได้ง่ายด้วยการเรียกใช้งานผ่านทาง single point และช่วยลดค่าอุปกรณ์ฮาร์ดแวร์ได้ เนื่องจากไคลเอ็นต์ทั้งภายในและภายนอกระบบเครือข่ายสามารถเข้ามาใช้งานเซิร์ฟเวอร์เดียวกันได้ ทำให้สามารถลดอุปกรณ์ที่ซ้ำซ้อนกันสำหรับให้บริการภายนอกและภายในระบบไปได้ รวมทั้งยังช่วยสร้างความ ปลอดภัยแก่ฐานข้อมูลเนื่องจากการให้บริการจะกระทำผ่านรีเวิร์สพร็อกซี่ ไม่สามารถเรียกใช้งานโดยตรงได้

ข้อเสียของรีเวิร์สพร็อกซี่

หากรีเวิร์สพร็อกซี่ไม่สามารถใช้งานได้และไม่สามรถกู้คืนได้ จะส่งผลให้ไม่สามารถใช้งานเว็บไซต์ตามไปด้วย ถ้าผู้บุกรุกสามารถเข้าไปยังรีเวิร์สพร็อกซี่ได้ ก็เท่ากับว่าผู้บุกรุกสามารถเข้าไปดูโครงสร้างของระบบเว็บเซิร์ฟเวอร์ได้ หรือถ้าเว็บเซิร์ฟเวอร์ถูกซ่อนอยู่ภายใต้ไฟร์วอลล์ผู้บุกรุก ก็จะสามารถเข้าไปยังระบบเครือข่ายภายในได้เช่นกัน ดังนั้นจึงมีความจำเป็นในการติดตั้งระบบรักษา

ความปลอดภัยให้แก่รีเวิร์สพร็อกซี่และเซิร์ฟเวอร์ทันทีที่เริ่มใช้งานหรือทำการทดสอบระบบทั้งหมด และติดตั้งซอฟต์แวร์สำหรับดักจับการบุกรุกให้แก่ เซิร์ฟเวอร์และระบบเครือข่ายด้วย

เมื่อมีการเปลี่ยนแปลงข้อมูลเกิดขึ้นหลายครั้งในรีเวิร์สพร็อกซี่และไฟร์วอลล์ทำให้การตอบสนอง request อาจช้าลง

เพื่อป้องการบุกรุกที่แอบแฝงเข้ามาทางโปรโตคอล HTTP ข้อมูลที่เป็น HTTP ที่ผ่านทางรีเวิร์สพร็อกซี่ต้องกลั่นกรองข้อมูลภายในก่อน เพื่อให้มั่นใจได้ว่าเป็น HTTP request จริงไม่ใช่ ผู้บุกรุก

ถ้าหากใช้ analysis-report tool กับเว็บเซิร์ฟเวอร์จะไม่สามารถรายงานปริมาณผู้เข้าใช้งานได้ เนื่องจาก log เก็บว่าทุก HTTP request ถูกส่งมาจากรีเวิร์สพร็อกซี่

รีเวิร์สพร็อกซี่ทำงานอย่างไร

ในการแสดงการทำงานของรีเวิร์สพร็อกซี่จะแสดงระบบที่จำลองขึ้น ดังนี้คือ เชื่อมต่อรีเวิร์ส พร็อกซี่กับ DMZ โดยให้เว็บเซิร์ฟเวอร์อยู่ภายในไฟร์วอลล์ใน subnet ภายใน website ชื่อ www.mysite.com มี static NAT address เป็น 10.0.1.1 และมี IP address จริงเป็น 192.168.1.1 website นี้ และ NAT address เชื่อมต่อกับรีเวิร์สพร็อกซี่เซิร์ฟเวอร์ ชื่อ myserver.mysite.com มี NAT address เป็น 10.0.2.1 และมี IP จริงเป็น 192.168.2.1

กำหนดกฎของไฟร์วอลล์ให้อนุญาตเฉพาะ แอดเดรส 192.168.1.1 สามารถ access 192.168.2.1 ผ่านทางพอร์ต 80 และ 443 และอนุญาตให้พอร์ต 80 และ 443 ส่งต่อไปยังแอดเดรส 192.198.1.1 ไฟล์ที่เก็บรายชื่อโฮสต์บนรีเวิร์สพร็อกซี่เก็บ entry ของโฮสต์ชื่อ myserver.mysite.com และ มี static NAT address เป็น 10.0.2.1

ก่อนใช้งานรีเวิร์สพร็อกซี่ต้องกำหนด prefix mapping ให้แก่รีเวิร์สพร็อกซี่ก่อน prefix mapping มี 2 ชนิด regular mapping และ reverse mapping

- regular mapping ใช้สำหรับบอกรีเวิร์สพร็อกซี่ว่า URL prefix ไหนถูกแทนที่และ บอก URL ปลายทางจริง
- reverse mapping จะแปลง URL prefix ไปเป็น URL ของรีเวิร์สพร็อกซี่บอกชื่อเว็บเซิร์ฟเวอร์จริงและอนุญาตให้ส่งต่อได้ ซึ่งเป็นส่วนที่สำคัญที่สุดของรีเวิร์สพร็อกซี่

ในการสื่อสารไคลเอ็นต์ส่ง HTTP GET request ไปที่ www.mysite.com (10.0.1.1) เมื่อไฟร์วอลล์ได้รับรีเควสต์นั้น นำไปตรวจกับ NAT rule ซึ่งกำหนดให้ ส่งรีเควสต์ไปยังรีเวิร์สพร็อกซี่ที่ไอพีแอดเดรส 192.168.1.1 เมื่อรีเวิร์สพร็อกซี่ได้รับรีเควสต์และตรวจสอบ URL prefix กับรีเวิร์สพร็อกซี่Mapping พบว่าต้องส่งรีเควสต์ต่อไปยัง myserver.mysite.com ดังนั้นรีเวิร์สพร็อกซี่จึงส่งรีเควสต์กลับไปยังไฟร์วอลล์เมื่อไฟร์วอลล์ได้รับรีเควสต์จะนำไปตรวจสอบกับ NAT rule พบว่าถูกส่งมาจาก 192.168.1.1 หรือรีเวิร์สพร็อกซี่ซึ่งได้รับอนุญาตให้สามารถเรียกใช้งาน 192.168.2.1 ทางพอร์ต 80 ได้ รีเควสต์จึงถูกส่งต่อไปยังเว็บเซิร์ฟเวอร์

เมื่อ myserver.mysite.com ประมวลผลรีเควสต์เสร็จและส่งผลลัพธ์กลับไปยังผู้ร้องขอซึ่ง ขณะนี้คือรีเวิร์สพร็อกซี่เมื่อรีเวิร์สพร็อกซี่ได้รับการตอบรับกลับมา จะนำไปตรวจกับ reverse mapping และเปลี่ยน URL และ HTTP header จาก myserver.mysite.com ไปเป็น www.mysite.com หลังจากแปลงข้อมูลแล้วข้อมูลผลลัพธ์จะถูกส่งกลับไปยังไคลเอ็นต์ซึ่งเป็นผู้ส่ง HTTP GET Request ทางด้านไคลเอ็นต์จะเห็นว่ารีเควสต์ถูกตอบสนองโดย www.mysite.com แต่แท้จริงแล้วถูกตอบสนองโดย myserver.mysite.com

โครงสร้างระบบ สำหรับใช้งานรีเวิร์พร็อกซี่

มีหลายสิ่งที่ต้องคำนึงในการติดตั้งรีเวิร์สพร็อกซี่สิ่งแรกซึ่งเป็นสิ่งที่สำคัญที่สุดคือเรื่องความปลอดภัย การใช้ static NAT address ในการอ้างถึงเซิร์ฟเวอร์มีความสำคัญ เมื่อ ผู้บุกรุก เข้ามาสำรวจใน ไซต์การใช้ไฟร์วอลล์ในการแปลง NAT address ไปเป็นแอดเดรสจริงของเซิร์ฟเวอร์ทำให้ ผู้บุกรุก ไม่ได้รับข้อมูลจริงใน โครงสร้างของระบบ

กฎของไฟร์วอลล์มีบทบาทหลักในรีเวิร์สพร็อกซี่ การกำหนดกฎของไฟร์วอลล์เพื่อกำหนดเส้นทางของ HTTP ไปยังรีเวิร์สพร็อกซี่และอนุญาตให้ส่ง HTTP request ไปยังเซิร์ฟเวอร์ เริ่มจากรีเวิร์สพร็อกซี่สร้าง single gateway ทำให้สามารถควบคุมเส้นทางของ HTTP จากจุดเดียว

มีเพียงชื่อโฮสต์ของพร็อกซี่เซิร์ฟเวอร์เท่านั้นที่เปิดเผยต่อภายนอกชื่อโฮสต์ของเว็บเซิร์ฟเวอร์อื่น ๆ ซึ่งรีเวิร์สพร็อกซี่ส่งต่อรีเควสต์ไปถูกเก็บใน host file ในเครื่อง รวมทั้งกำหนด static NAT address ให้ชื่อโฮสต์แต่ละอันไว้ใน host file ด้วย

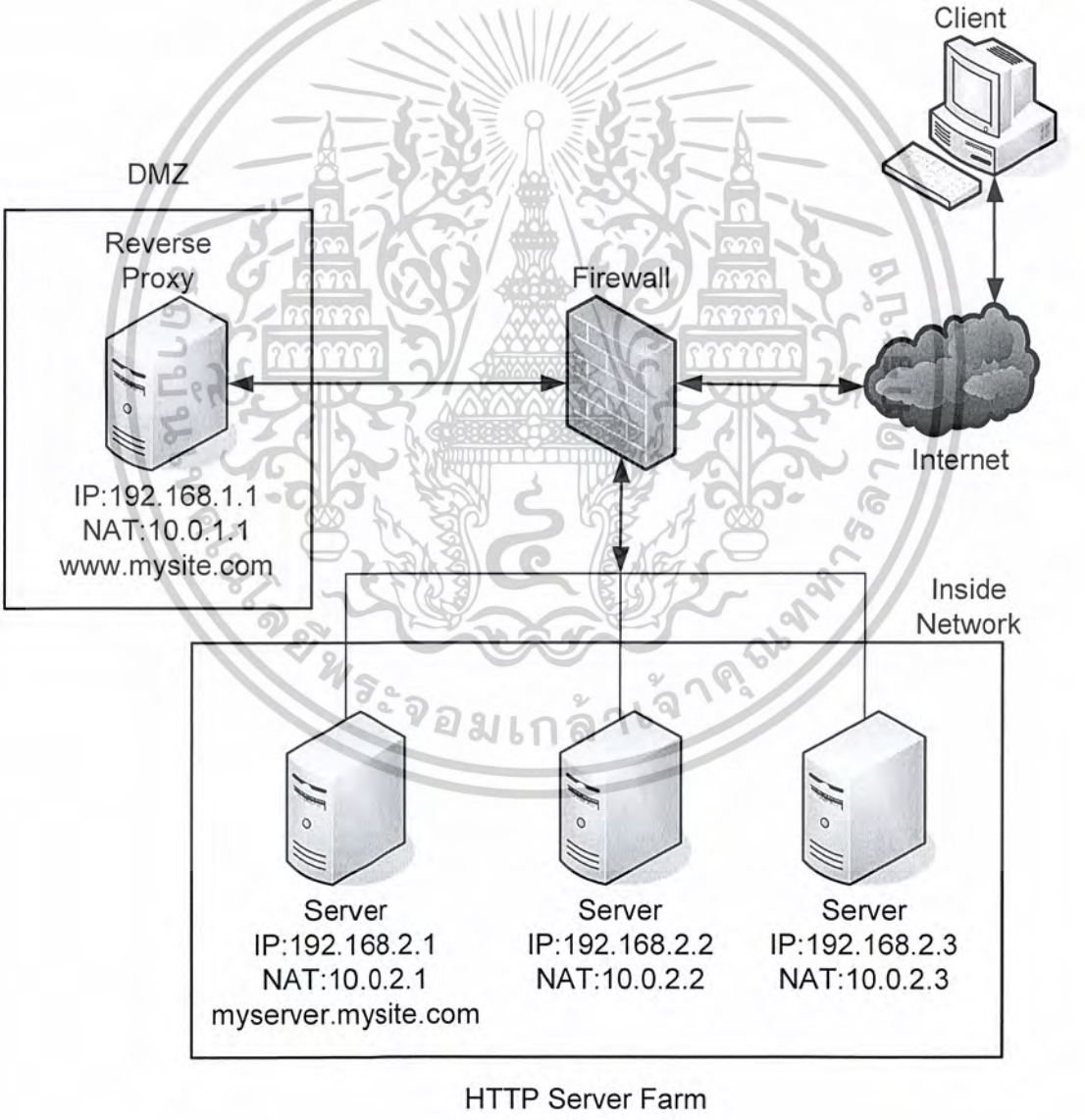
ต้องติดตั้งการตรวจจับการบุกรุกให้แก่รีเวิร์สพร็อกซี่และเซิร์ฟเวอร์ทุกเครื่องที่รับ HTTP request และต้องหมั่นตรวจดู log ด้วย สำหรับพร็อกซี่เซิร์ฟเวอร์สามารถใช้ Apache, Microsoft Proxy, Squid หรือพร็อกซี่อื่นที่รองรับ regular mapping และ reverse mapping

การติดตั้งรีเวิร์สพร็อกซี่มีอย่างน้อย 2 รูปแบบดังนี้ วิธีแรก คือออกแบบให้ เซิร์ฟเวอร์อยู่ในเครือข่ายภายใน ภายในไฟร์วอลล์และรีเวิร์สพร็อกซี่อยู่บน DMZ เว็บเซิร์ฟเวอร์ต้องอยู่ใน subnet เฉพาะเพื่อแยกออกจากส่วนอื่นในเครือข่ายภายใน ถ้าเซิร์ฟเวอร์เป็น Windows NT หรือ Windows 2000 และใช้ Windows authentication ต้องแยกโดเมนเพื่อป้องกันการบุกรุก หากไคลเอ็นต์ใน เครือข่ายภายในต้องการรับบริการจากเว็บเซิร์ฟเวอร์ต้องกำหนด user id ใหม่และพาสเวิร์ดใหม่สำหรับโดเมนนี้ ด้วย เพื่อป้องกันไม่ให้ ผู้บุกรุก เข้ามาในเครือข่ายภายใน แม้ผู้บุกรุก จะมี user idและพาสเวิร์ดของผู้ใช้เครือข่ายภายนอกก็ตาม

ประโยชน์ ที่จะได้รับคือไคลเอ็นต์จากทั้งภายในและภายนอกเครือข่ายสามารถรับบริการจากเซิร์ฟเวอร์ชุดเดียวกันได้ เป็นการลดค่าอุปกรณ์ฮาร์ดแวร์ลง ไม่ต้อง replicate content จะช่วยลด bandwidth ได้อีกด้วย ค่าใช้จ่ายในการดูแลระบบลดลงเนื่องจากมีเซิร์ฟเวอร์เพียงชุดเดียวและสามารถดูแลและจัดการได้เหมือนเป็น เซิร์ฟเวอร์ภายในทั่วไปผู้ดูแลระบบจึงทำงานได้ง่าย

ข้อเสียคือ ถ้า รีเวิร์สพร็อกซี่ถูกโจมตี และผู้บุกรุกสามารถเข้าถึงเว็บเซิร์ฟเวอร์ได้ ผู้บุกรุกจะสามารถเข้าถึงเครือข่ายภายในได้เช่นกัน

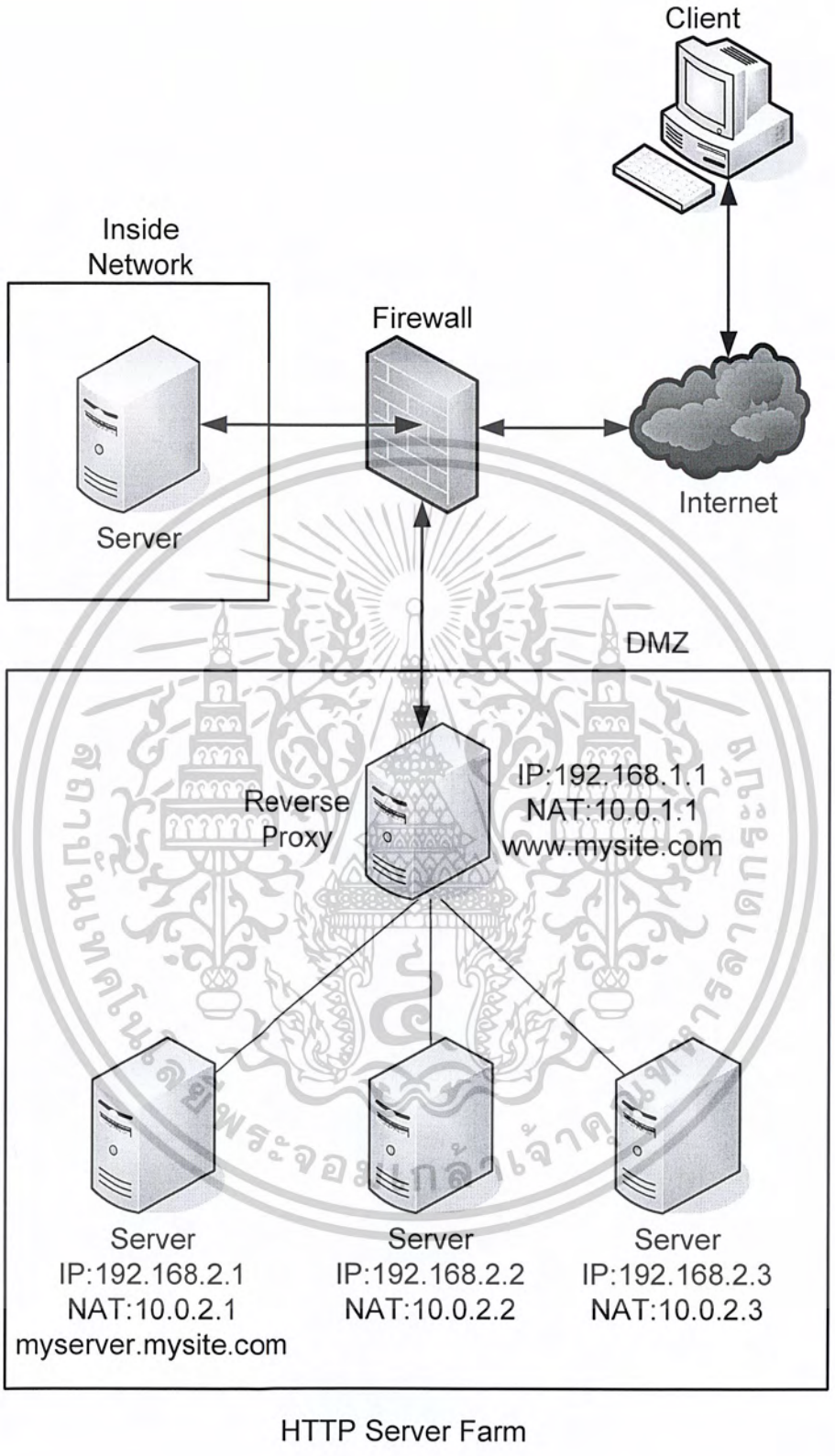
แม้ว่าอาจเกิดความเสี่ยงที่ผู้บุกรุกจะสามารถเข้าถึงเครือข่ายภายในได้ แต่เมื่อพิจารณาระบบความปลอดภัยของรีเวิร์สพร็อกซี่ แล้วมีเพียงรีเวิร์สพร็อกซี่เท่านั้นที่สามารถเรียกใช้งานโดยตรงไปยังเซิร์ฟเวอร์ได้ และในกรณีที่จะเรียกใช้งานเซิร์ฟเวอร์ HTTP request ต้องผ่าน ระบบตรวจจับการบุกรุก ถึง 4 จุดด้วยกัน โดยเริ่มจาก HTTP request จากไคลเอ็นต์ถูกส่งผ่านไฟร์วอลล์ , รีเวิร์สพร็อกซี่ , ไฟร์วอลล์ และท้ายสุดคือเซิร์ฟเวอร์ ถ้ารวมถึงขั้นตอนการตรวจดูเนื้อหาภายในของรีเควสต์ด้วยจะเป็น 5 จุดด้วยกัน ซึ่งในกรณีนี้การบริหารและจัดการหลักจะต้องอยู่ภายในเครือข่ายของเซิร์ฟเวอร์เท่านั้น ทำให้ User id และพาสเวิร์ดของ ผู้ดูแลระบบจะไม่ถูกเปิดเผยจากการดักฟังของผู้บุกรุก เมื่อระบบถูกบุกรุก ผู้บุกรุกจึงเข้าถึงได้เพียงเซิร์ฟเวอร์บน subnet เดียวกันได้เท่านั้น เมื่อพิจารณาข้อดีทั้งหมดแล้วลักษณะโครงสร้างนี้จึงเป็นทางเลือกที่น่าสนใจ



รูปที่ 2-7 แสดงการเชื่อมต่อกับ server farm แบบที่ 1

วิธีที่2 ออกแบบให้เว็บเซิร์ฟเวอร์และพร็อกซี่อยู่บน DMZ กำหนด NAT address ให้แก่เว็บเซิร์ฟเวอร์ไม่เก็บไว้ในภายนอก DNS แต่เก็บไว้ใน host file เพื่อใช้ค้นหาไอพีแอดเดรสจากชื่อของโฮสต์ หากมีเซิร์ฟเวอร์ที่เป็น Windows NT หรือ Windows 2000 ต้องติดตั้งโดเมนภายนอกด้วยเพื่อทำ authentication และไฟร์วอลล์จะเป็นผู้ดูแลการส่ง HTTP request ไปยังรีเวิร์สพร็อกซี่อีกเช่นกัน

ส่วนที่แตกต่างจากวิธีแรกคือรีเวิร์สพร็อกซี่สามารถส่ง HTTP request ไปให้เซิร์ฟเวอร์ได้โดยตรง ประโยชน์คือเครือข่ายภายในแยกเครือข่ายภายในอย่างเด็ดขาด ผู้บุกรุกจึงไม่สามารถเข้ามายังเครือข่ายภายในได้ แต่ไคลเอ็นต์ยังคงเรียกใช้งาน ผ่าน single point เช่นเดิม ข้อเสียคือต้องมีเซิร์ฟเวอร์ 2 ชุดสำหรับให้บริการไคลเอ็นต์ทั้งภายนอกและภายในซึ่งเพิ่มค่าใช้จ่ายทางด้านอุปกรณ์ฮาร์ดแวร์และภาระการดูแลระบบเพิ่มมากขึ้นอีกด้วย



รูปที่ 2-8 แสดงการเชื่อมต่อกับ server farm แบบที่ 2

2.4 โพรโตคอลเอชทีทีพี (HTTP : HyperText Transfer Protocol)

โพรโตคอลนี้สร้างขึ้นสำหรับบริการที่เรียกว่า WWW (World Wide Web) ในเครือข่ายอินเตอร์เน็ตโดยเฉพาะ โพรโตคอลนี้จะเป็นตัวกำหนดวิธีการส่งข้อมูลหรือไฟล์ระหว่างเครื่องคอมพิวเตอร์ลูกข่าย (Client) และ เครื่องแม่ข่าย (Server) รวมถึงกำหนดกฎระเบียบในการติดต่อด้วย โพรโตคอลนี้ถูกออกแบบมาให้มีความกระทัดรัด สามารถทำงานได้รวดเร็ว มีการระบวนการทำงานที่ไม่ซับซ้อน และมีคำสั่งที่ใช้งานไม่มากนัก แต่สามารถรองรับข้อมูลได้ทุกแบบ ไม่ว่าจะเป็นข้อมูลทั่วไปที่เข้ารหัสแบบ MIME หรือข้อมูลที่เป็นกราฟิก เช่น ไฟล์ที่เป็น GIF หรือ JPEG เป็นต้น

2.4.1 วิธีการติดต่อของโพรโตคอล HTTP

โพรโตคอล HTTP อยู่บนพื้นฐานของระบบเครือข่ายไคลเอนต์ / เซิร์ฟเวอร์ (Client / Server) ที่ต้องมีการร้องขอรับบริการจากไคลเอนต์ (request) และการตอบสนองหรือการให้บริการของเซิร์ฟเวอร์ (response) จึงสามารถแบ่งการทำงานออกเป็น 2 ด้านคือ ด้านเว็บเซิร์ฟเวอร์ และด้านไคลเอนต์ โดยไคลเอนต์จะติดต่อเข้ามายังเซิร์ฟเวอร์และอ้างถึงแอดเดรสของเซิร์ฟเวอร์โดยใช้รูปแบบของURLส่วนด้านเซิร์ฟเวอร์จะส่งข้อมูลกลับมาในรูปแบบที่เป็นภาษา HTML(HyperText Markup Language) โดยที่โพรโตคอล HTTP ใช้วิธีการเข้ารหัสในแบบ MIME เป็นมาตรฐานของการทำงาน โพรโตคอลนี้ถูกออกแบบมาให้สามารถรับส่งข้อมูลผ่าน Proxy หรือ Firewall ต่าง ๆ ได้ โดยอาศัยการเชื่อมต่อผ่านทางโพรโตคอล TCP/IP อีกทีหนึ่ง โดยใช้พอร์ตหมายเลข 80 เป็นช่องทางมาตรฐานในการติดต่อ ในทางปฏิบัติจะใช้พอร์ตหมายเลขอื่นก็ได้ ในปัจจุบันเว็บบราวเซอร์ทั่วไปจะกำหนดค่ามาตรฐานไว้ที่พอร์ต 80 ดังนั้นหากมีการกำหนดไว้ที่พอร์ตอื่น จะทำให้เกิดความลำบากต่อผู้ใช้ที่ต้องระบุหมายเลขพอร์ตลงใน URL ด้วย

ด้วยเหตุที่การทำงานของโพรโตคอล HTTP เป็นแบบไคลเอนต์และเซิร์ฟเวอร์ ดังนั้นการติดต่อสื่อสารใดๆผ่านโพรโตคอลนี้จำเป็นต้องมีเครื่องตัวแม่กับตัวลูก การสื่อสารจะสมบูรณ์ได้ การติดต่อกันระหว่างไคลเอนต์ไปยังเซิร์ฟเวอร์ผ่านโพรโตคอล HTTP มีขั้นตอนดังนี้

ขั้นแรก : Open Socket

ไคลเอนต์ จะสร้างการเชื่อมต่อ (Connection) กับเซิร์ฟเวอร์ผ่านซ็อกเก็ต (Socket)

ขั้นที่สอง : Request

ไคลเอนต์ส่งคำร้องขอข้อมูลไปยังเซิร์ฟเวอร์

ขั้นที่สาม : Information Transfer

เซิร์ฟเวอร์จะไปหาข้อมูลที่ไคลเอนต์ต้องการ

ขั้นที่สี่ : Response

เซิร์ฟเวอร์ส่งข้อมูลตอบสนอง (Response) กลับมายังไคลเอนต์เสมอ

ขั้นสุดท้าย : Close Socket

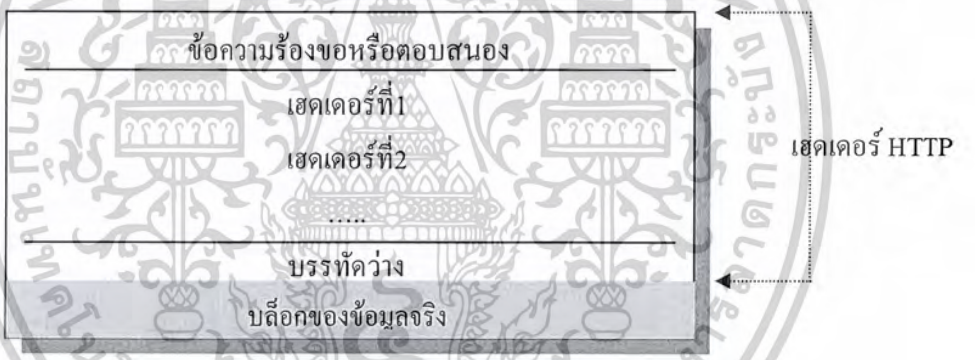
ปลดการเชื่อมต่อของซ็อกเก็ตของทั้งสองฝั่งออก

ด้วยการทำงานของโพรโตคอล HTTP ที่มีการเชื่อมต่อในระยะเวลาเพียงสั้น หรือที่เรียกว่าเป็นโพรโตคอลแบบ Connectionless ในลักษณะดังกล่าว ทำให้ในช่วงเวลาหนึ่ง ๆ เซิร์ฟเวอร์ที่ให้บริการ WWW สามารถรองรับไคลเอนต์ได้จำนวนมากพร้อม ๆ กัน เพราะไม่มีใครได้ทำการเชื่อมต่ออย่างถาวร

ในการร้องขอรับบริการจากไคลเอนต์ และการตอบสนองหรือการให้บริการจากเซิร์ฟเวอร์นั้น ย่อมต้องมีการรับส่งข้อมูลระหว่างกัน แต่ข้อมูลที่รับส่งกันในแต่ละครั้งไม่ได้มีเฉพาะข้อมูลเพียงอย่างเดียว แต่ละฝ่ายจะมีส่วนเฮดเดอร์ HTTP (HTTP header) เข้าไปในส่วนต้นของข้อมูลที่รับ-ส่งกันด้วย ซึ่งเฮดเดอร์ HTTP จะเป็นตัวบอกว่าข้อมูลที่ส่งมานี้เป็นข้อมูลอะไร เป็นข้อมูลการร้องขอจากไคลเอนต์ หรือเป็นข้อมูลตอบสนองจากเซิร์ฟเวอร์

2.4.2 โครงสร้างของโพรโตคอล HTTP

โครงสร้างของข้อมูล HTTP จะแบ่งออกเป็น 2 ส่วนใหญ่ ๆ คือ ส่วนเฮดเดอร์ หรือเรียกว่า metadata จะเป็นส่วนเก็บข้อมูลที่จำเป็นต้องใช้ภายในโพรโตคอล ส่วนที่สองเป็นส่วนของข้อมูลจริงที่ต้องการรับส่ง ดังแสดงในรูปที่ 2-3



รูปที่ 2-9 โครงสร้างของข้อมูลที่ส่งผ่านโพรโตคอล HTTP

2.4.2.1 เฮดเดอร์เอชทีทีพี (HTTP Header)

ดังรูปที่ 2-3 เฮดเดอร์เอชทีทีพี (HTTP Header) ประกอบด้วย 2 ส่วน คือ ส่วนข้อมูลร้องขอหรือตอบสนอง และส่วนเฮดเดอร์ย่อย

2.4.2.1.1 ส่วนข้อมูลร้องขอหรือตอบสนอง

ส่วนนี้เป็นส่วนในการแยกว่าเป็นข้อความตอบสนองจากเซิร์ฟเวอร์ หรือข้อความร้องขอจากไคลเอนต์ และรายละเอียด ดังต่อไปนี้

2.4.2.1.1.1. ข้อความร้องขอ (request)

จากรูปที่ 2-3 ในส่วนข้อความการร้องขอ(บรรทัดแรก) จะประกอบด้วย 3 ส่วน คือ

1. วิธีการร้องขอ หรือที่เรียกว่า "เมธอด"
2. ไดเรกทอรีและชื่อไฟล์ที่ต้องการจากเซิร์ฟเวอร์
3. เวอร์ชันของ HTTP ที่ไคลเอนต์ใช้อยู่

โดยที่แต่ละส่วนจะถูกแบ่งด้วยช่องว่าง(space)เขียนรูปแบบในการเขียนข้อความบรรทัดแรกของเฮดเดอร์ HTTP ได้ดังนี้

<Method> /path/file HTTP/x.x

สังเกตว่าการขอข้อมูลจากเซิร์ฟเวอร์ จะระบุเฉพาะไดเรกทอรี และชื่อไฟล์ที่ต้องการ ไม่ต้องบอกชื่อโฮสต์และโดเมนเนมเพราะได้มีการสร้างการเชื่อมต่อกับโฮสต์ในโพรเซสก่อนหน้านี้ไปแล้ว การร้องขอข้อมูลจึงไม่ต้องระบุชื่อโฮสต์ซ้ำอีกครั้ง

นอกจากคำร้องขอที่อยู่ในบรรทัดแรกแล้ว ยังมีข้อมูลอื่น ๆ ที่ส่งไปให้กับเซิร์ฟเวอร์ด้วย คือข้อมูลในส่วนที่สอง ซึ่งเรียกว่า "เฮดเดอร์" (header) ข้อมูลในเฮดเดอร์แต่ละเว็บบราวเซอร์แต่ละเวอร์ชันอาจจะไม่เหมือนกัน ซึ่งข้อมูลเฮดเดอร์นี้จะบอกรายละเอียดของผู้ส่งว่ามีอะไรบาง เฮดเดอร์นี้จะบอกให้เซิร์ฟเวอร์ทราบได้ว่าข้อความร้องขอถูกส่งมาจากใคร และสามารถนำไปใช้เพื่อประโยชน์ในเรื่องอื่น ๆ ต่อไปได้

จากโครงสร้างข้อมูลที่รับส่งระหว่างไคลเอนต์กับเซิร์ฟเวอร์ในรูปที่ 2-3 หลังจากเฮดเดอร์รายการสุดท้ายแล้ว จะมีบรรทัดว่าง หลังจากนั้นจะเป็นส่วนของบล็อกข้อมูล ทั้งนี้หากเป็นการร้องขอจากไคลเอนต์ ก็จะขึ้นอยู่ว่าใช้เมธอดใดในการร้องขอ หากเป็นเมธอดGET ก็ไม่จำเป็นต้องมีข้อมูลอะไรในส่วนนี้ เนื่องจากเซิร์ฟเวอร์จะไม่สนใจข้อมูลในส่วนนี้ ทั้งนี้เนื่องจากรูปแบบของโพรโตคอลนั่นเอง จะกล่าวละเอียดต่อไป

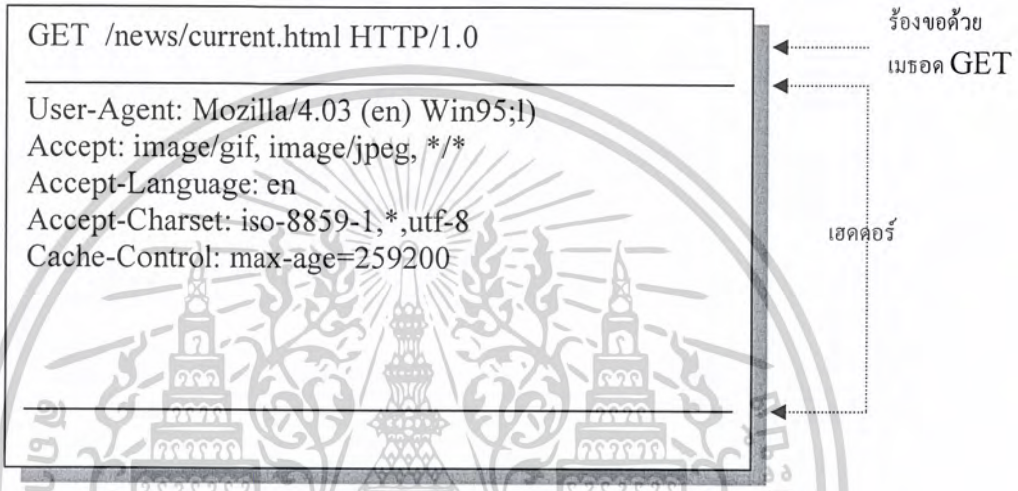
เมื่อไคลเอนต์เชื่อมต่อกับเซิร์ฟเวอร์เรียบร้อยแล้ว ไคลเอนต์จะเป็นฝ่ายส่งข้อมูลการร้องขอไปยังเซิร์ฟเวอร์ ซึ่งการร้องขอไปยังเซิร์ฟเวอร์นี้สามารถทำได้หลายวิธี ทั้งนี้ทั้งนั้นก็ขึ้นอยู่กับเวอร์ชันของโพรโตคอล HTTP ที่ใช้ หากเป็นเวอร์ชัน 1.0 จะมีวิธีการร้องขอมาตรฐาน 3 วิธี คือ GET, HEAD และ POST แต่หากเป็นโพรโตคอล HTTP เวอร์ชัน 1.1 จะมีวิธีการร้องขอเพิ่งจากเวอร์ชัน 1.0 อีกหลายวิธี เช่น OPTIONS, PUT, DELETE หรือ TRACE เป็นต้น ดังนั้นการที่เราจะเลือกว่าจะใช้โพรโตคอล HTTP เวอร์ชันไหน ต้องขึ้นอยู่กับเซิร์ฟเวอร์ และ ไคลเอนต์ที่จะทำงานด้วย คือ หากว่าเซิร์ฟเวอร์สนับสนุนการทำงานของ HTTP 1.1 แล้ว วิธีการร้องขอก็สามารถใช้ของเวอร์ชัน 1.1 ได้

เพื่อไม่ให้เกิดปัญหาในการทำงาน จึงควรใช้โพรโตคอล HTTP เวอร์ชัน 1.0 ซึ่งเว็บเซิร์ฟเวอร์ส่วนใหญ่ในอินเตอร์เน็ตจะสามารถรองรับการร้องขอจากเวอร์ชันนี้ โดยโพรโตคอล HTTP เวอร์ชัน 1.0 มีวิธีการร้องขออยู่ 3 วิธีด้วยกัน โดยมีรายละเอียดดังนี้

1. การร้องขอด้วยเมธอด GET
   มีรูปแบบดังนี้

   GET /path/file HTTP/x.x

เป็นการร้องขอให้เซิร์ฟเวอร์ส่งไฟล์มาให้ หรือ เป็นการร้องขอโดยมีการส่งข้อมูลจากทางไคลเอนต์ไปให้ด้วยก็ได้ ดูตัวอย่างประกอบในรูปที่ 2-4



รูปที่ 2-10 ตัวอย่างข้อความร้องขอด้วยเมธอด GET

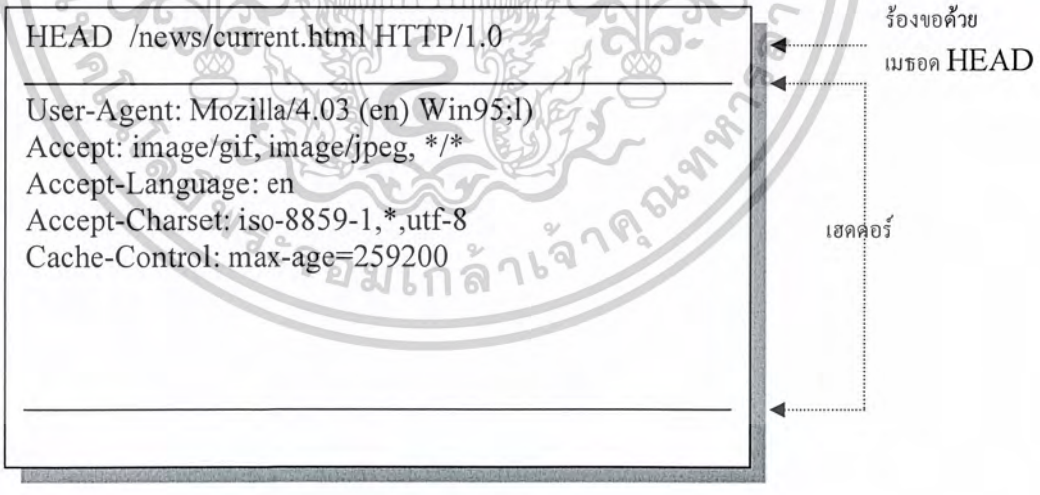
นอกจากข้อความร้องขอด้วยเมธอด GET ใช้สำหรับการร้องขอข้อมูลจากเซิร์ฟเวอร์แล้ว ข้อความร้องขอด้วยเมธอด GET นี้ยังสามารถใช้สำหรับส่งข้อมูลไปยังเครื่องเซิร์ฟเวอร์ได้อีกด้วย โดยข้อมูลที่ต้องการส่งให้กับเซิร์ฟเวอร์นั้นจะต่อท้าย URL โดยมีเครื่องหมาย ? กั้นระหว่าง URL กับข้อมูลนั้น ลักษณะข้อมูลจะประกอบด้วยตัวแปร และค่าของตัวแปรนั้น โดยเขียนต่อในลักษณะ <Variable Name>=<Variable Value>&<Variable Name>=<Variable Value>&..... แต่มีข้อจำกัดด้านขนาดของข้อมูลที่ส่งไปให้เซิร์ฟเวอร์ด้วยเมธอดนี้ เนื่องจากให้ส่งได้ครั้งละไม่เกิน 256 ตัวอักษร(นับจากที่เข้ารหัสแล้ว) ความยาวทั้งหมดเริ่มนับจากชื่อ ไดเรกทอรีเป็นต้นไป แต่ความยาวนี้ขึ้นอยู่กับระบบปฏิบัติการอีกทีหนึ่ง

2. การร้องขอด้วยเมธอด HEAD
มีรูปแบบดังนี้

HEAD /path/file HTTP/x.x

เป็นการร้องขอเพื่อถามเซิร์ฟเวอร์ว่ามีไฟล์ที่ต้องการอยู่ในเซิร์ฟเวอร์หรือไม่ (ถามเฉย ๆ ไม่ต้องการให้เซิร์ฟเวอร์ส่งไฟล์จริงมาให้) ซึ่งมีประโยชน์สำหรับการตรวจสอบว่ามีไฟล์ที่ต้องการอยู่ในเซิร์ฟเวอร์หรือไม่ หรือใช้ตรวจสอบความสมบูรณ์ของลิงค์ก็ได้ รหัสตอบสนองในบรรทัดสถานะจึงอาจเป็น 200 (มีไฟล์ที่ต้องการ) หรือ 404 (ไม่มีไฟล์ที่ต้องการ) และข้อมูลอื่นส่งเพิ่มเติมมาในเฮดเดอร์ด้วย เช่น วันที่ปรับปรุงแก้ไขไฟล์ครั้งสุดท้าย (Last Modified) เป็นต้น

การจะตรวจสอบว่ามีไฟล์นั้นอยู่ที่เซิร์ฟเวอร์หรือไม่ อาจใช้วิธีการร้องขอด้วยเมธอด GET แล้วใช้การเช็คผลการตอบสนอง แต่การตอบสนองจากการร้องขอด้วยเมธอด HEAD จะไม่มีการส่งเนื้อหาของไฟล์มาให้ (ไม่มีบล็อกข้อมูล) ดังนั้นใช้เมธอด HEAD จึงทำงานได้คำตอบเร็วกว่า มักจะใช้เมธอดนี้ในการตรวจสอบกับทางเซิร์ฟเวอร์ หากมีการปรับปรุงไฟล์นั้น จึงจะมีการ download มาทับไฟล์เดิมในเครื่อง (ร้องขอไฟล์นั้นอีกครั้งด้วยเมธอด GET) ดูตัวอย่างประกอบในรูปที่ 2-5



รูปที่ 2-11 ตัวอย่างข้อความร้องขอด้วยเมธอด HEAD

3. การร้องขอด้วยเมธอด POST
มีรูปแบบดังนี้

POST /path/file HTTP/x.x

เป็นการร้องขอให้เซิร์ฟเวอร์รับข้อมูลจากไคลเอนต์เพื่อนำไปประมวลผลหรือนำไปเก็บในฐานข้อมูลต่อไป โดยมีเงื่อนไขดังนี้

- ข้อมูลที่จะส่งไปให้เซิร์ฟเวอร์จะอยู่ภายในบล็อกข้อมูล ดังนั้นจึงต้องมีเฮดเดอร์เพื่อบอกรายละเอียดของข้อมูลในบล็อกข้อมูลแนบไปด้วย
- /path/file คือ ชื่อโปรแกรม CGI ในเซิร์ฟเวอร์ที่จะทำหน้าที่รับข้อมูลไปประมวลผล
- ข้อความตอบสนองที่เซิร์ฟเวอร์จะส่งกลับให้ไคลเอนต์ จะได้มาจากกการทำงานของโปรแกรม CGI ในเซิร์ฟเวอร์ ดังนั้น CGI ที่รับข้อมูลไปจึงต้องทำหน้าที่ส่งข้อความตอบกลับให้ไคลเอนต์

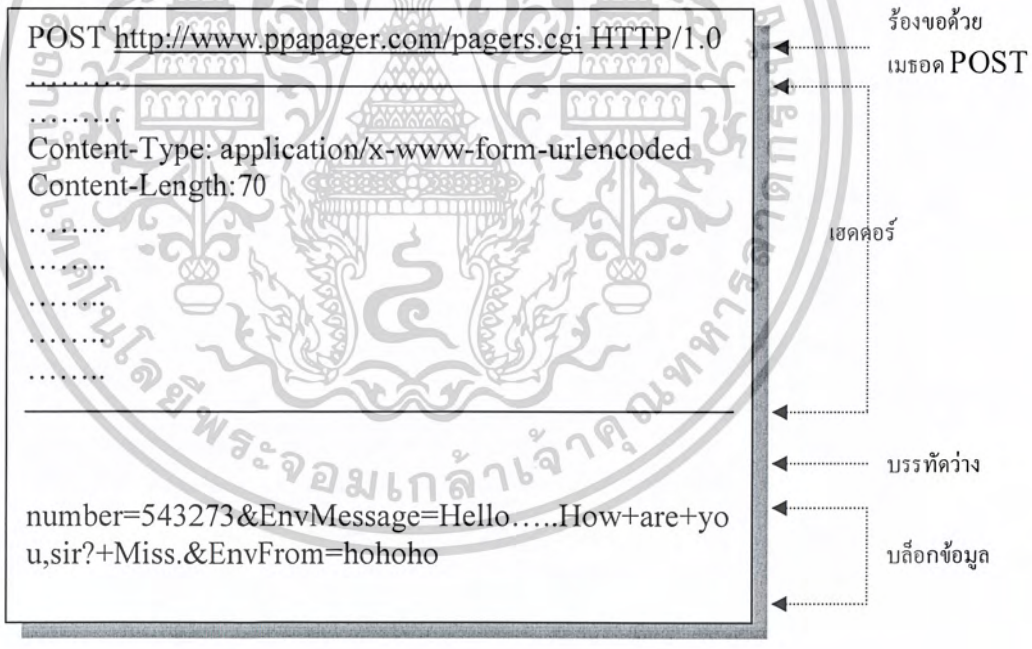
โดยส่วนใหญ่การส่งข้อมูลจากฟอร์มในเว็บเพจไปประมวลผลด้วย CGI ในเซิร์ฟเวอร์จะใช้เมธอดนี้มากที่สุด ความจริงแล้วการส่งข้อมูลไปยังเซิร์ฟเวอร์สามารถใช้เมธอด GET ก็ได้ แต่มีข้อจำกัดเรื่องความยาวของข้อมูลที่จะส่งไปให้เซิร์ฟเวอร์ ถ้าใช้เมธอด POST เพื่อร้องขอส่งข้อมูลไปยังเซิร์ฟเวอร์ เฮดเดอร์ที่ชื่อ Content-Type จะถูกกำหนดให้เป็น application/x-www-form-urlencoded เพื่อบอกแก่เซิร์ฟเวอร์ว่าข้อมูลที่ส่งไปให้มีการเข้ารหัส และเฮดเดอร์ Content-Length จะใช้สำหรับบอกความยาวของข้อมูลที่เข้ารหัสแล้ว เพราะข้อมูลที่กรอกผ่านอินพุตในฟอร์มจะถูกเว็บบราวเซอร์เข้ารหัสก่อนส่งเสมอ การเข้ารหัสนี้เรียกว่า URL-encoded (ดูการเข้ารหัสในรูปที่ 2-6) ซึ่งมีรายละเอียดดังนี้

- แปลงตัวอักษรหรือเครื่องหมายบางตัวให้อยู่ในรูป %xx โดยที่ xx จะเป็นค่ารหัสแอสกีของตัวอักษรนั้น ตัวอักษรที่ต้องมีการแปลง เช่น =, &, % และ + เพราะเป็นเครื่องหมายที่ใช้เป็นตัวแบ่งแยกข้อมูลที่จะส่งไปให้เซิร์ฟเวอร์ ดังรูปที่ 2-6
- เปลี่ยนช่องว่าง (Space) ทุกตัวเป็นเครื่องหมายบวก (+)
- รวมชื่อตัวแปรและค่าตัวแปรเข้าด้วยกัน โดยคั่นกลางด้วยเครื่องหมาย = และคั่นระหว่างตัวแปรด้วยเครื่องหมาย &

อักขระ	รหัส (%xx)
%	%25
&	%26
‘	%27
+	%2B
=	%3D
?	%3F

รูปที่ 2-12 URL-encoded

ตัวอย่างข้อความร้องขอด้วยเมธอด POST ที่เว็บบราวเซอร์สร้างขึ้น เพื่อส่งข้อมูลจากฟอร์มในเว็บเพจไปให้แก่ CGI ที่ชื่อ pagerkara.cgi ในไดเรกทอรี / ของเซิร์ฟเวอร์ http://www.ppapager.com รับไปทำงาน ดังรูปที่ 2-7



รูปที่ 2-13 ตัวอย่างข้อความร้องขอด้วยเมธอด POST

2.4.1.1.2 ข้อความตอบสนอง (respond) และสถานะการทำงาน

โพรโตคอล HTTP ได้กำหนดรหัสแสดงสถานะการทำงานองโพรโตคอลไว้ โดยแบ่งกลุ่มของรหัสสถานะออกไว้เป็น 5 กลุ่ม ดังแสดงในรูปที่ 2-8 ดังนี้

รหัสสถานะ	ประเภท	รายละเอียด
100-199	Informational	เป็นรหัสสถานะที่เปิดให้โปรแกรมประยุกต์ต่าง ๆ กำหนดใช้งานได้เอง
200-299	Successful	กลุ่มรหัสที่แสดงว่าการทำงานสำเร็จ
300-399	Redirection	กลุ่มรหัสนี้จะใช้ภายในโพรโตคอลHTTP เองโดยเป็นการทำงานที่ต่อเนื่องมาจากโพรเซสก่อนหน้านี้ ซึ่งไคลเอนต์เป็นผู้สั่งงาน
400-499	Client Error	ใช้แสดงปัญหาที่เกิดขึ้นทางฝั่งไคลเอนต์
500-599	Server Error	ใช้แสดงปัญหาที่เกิดขึ้นทางฝั่งเซิร์ฟเวอร์

รูปที่ 2-14 กลุ่มของรหัสสถานะการทำงานของโพรโตคอล HTTP

รหัสแสดงสถานะในแต่ละตัว จะนำหน้าด้วยตัวเลข 3 หลัก และตามด้วยตัวอักษร ซึ่งรหัสในกลุ่ม 100-199 จะเปิดกว้างให้ผู้ที่พัฒนาโปรแกรมประยุกต์สามารถกำหนดค่าขึ้นมาใช้งานได้เอง ส่วนรายละเอียดของรหัสในกลุ่มอื่น ๆ จะแสดงในรูปที่ 2-9 ดังต่อไปนี้

รหัสสถานะ	รายละเอียด
100 Continue (1.1)	ใช้ในกรณีที่บราวเซอร์อยู่ในระหว่างส่ง Request แต่ยังส่งไม่หมด แต่เซิร์ฟเวอร์ต้องการให้ทราบว่าได้รับ Request แล้วให้ส่งส่วนที่เหลือต่อไป
101 Switching Protocol (1.1)	ใช้ร่วมกับเฮดเดอร์ Upgrade กรณีที่ต้องการเปลี่ยนไปใช้โพรโตคอลอื่นที่ความสามารถสูงกว่า เช่น HTTP/2.0 ซึ่งอาจจะมีในอนาคต
200 OK	การทำงานสำเร็จเรียบร้อย
201 Created	คำสั่ง POST ทำงานเสร็จสมบูรณ์
202 Accepted	ได้รับคำสั่งให้ทำงานเรียบร้อย แต่ไม่ต้องมีการตอบกลับ
203 Non-Authoritative(1.1)	การร้องขอประสบผลสำเร็จ
204 No Content	ทำงานตามคำสั่งเรียบร้อย แต่ไม่ต้องการแสดงข้อความใด ๆ บนหน้าจอ
205 Reset Content (1.1)	เซิร์ฟเวอร์ได้รับข้อมูลเรียบร้อยแล้ว และบอกให้บราวเซอร์ลบข้อความที่กรอกในแบบฟอร์มเดิมออก เพื่อนสะดวกในการกรอกข้อมูลถัดไป
206 Partial Content (1.1)	เซิร์ฟเวอร์ได้รับข้อมูลบางส่วนเรียบร้อยแล้ว
300 Multiple Choice	ถ้าค้นหาและพบแหล่งข้อมูลที่ต้องการหลายแห่ง เซิร์ฟเวอร์จะตอบกลับไปทั้งหมดเพื่อให้ไคลเอนต์สามารถเลือกแหล่งข้อมูลที่ต้องการเองได้
301 Moved Permanently	URL ที่ร้องขอได้ถูกย้ายไปที่อื่นแล้ว ดังนั้นการร้องขอใช้งาน กับ URL จะต้องเปลี่ยนเป็นแอดเดรสใหม่
302 Moved Temporarily	URL ที่ร้องขอมาได้ถูกย้ายไปที่อื่นชั่วคราว
303 See Other (1.1)	ใช้กรณีที่ต้องการบอกให้ทราบว่าสิ่งที่ต้องการอยู่ใน URI อื่น ซึ่งบราวเซอร์สามารถใช้ GET เพื่อเรียกดูเอกสารนั้น ๆ ได้
304 Not Modify	ใช้แสดงสถานะเมื่อใช้คำสั่ง GET ที่กำหนดเงื่อนไขเฉพาะเว็บไซต์ที่มีการเปลี่ยนแปลง ส่วนเว็บไซต์ที่ไม่มีการเปลี่ยนแปลงจะแสดงด้วยสถานะนี้
305 Use Proxy (1.1)	บอกให้บราวเซอร์ทราบว่าเอกสารที่ต้องการมีอยู่ใน Proxy ซึ่ง URL ของ Proxy จะกำหนดใน Location
400 Bad Request	คำสั่งจากไคลเอนต์ไม่ถูกต้อง
401 Unauthorized	ปฏิเสธการทำงานจากไคลเอนต์ที่ไม่ได้รับอนุญาต
403 Forbidden	เซิร์ฟเวอร์ไม่อนุญาตให้ใช้งาน หรือไคลเอนต์มีสิทธิ์ในการใช้งานเพียงพอ
404 Not Found	ไม่พบเว็บเซิร์ฟเวอร์ตาม URL ที่กำหนด
405 Method Not Allowed (1.1)	Method ที่ใช้ ไม่ได้รับอนุญาต กรณีนี้เซิร์ฟเวอร์จะระบุ Allow เพื่อบอกให้ทราบว่าอนุญาตให้ใช้ Method ใดบ้าง
406 Not Acceptable (1.1)	ข้อมูลที่ต้องการเป็นข้อมูลที่บราวเซอร์ไม่สามารถเข้าใจได้ เนื่องจากไม่อยู่ในรายการ Accept ที่ระบุใน Request Header

รูปที่ 2-15 รหัสสถานะ

รหัสสถานะ	รายละเอียด
407 Proxy Authentication ProxyAuthenticate (Unauthorized)	เหมือนกับ 401 แต่ต้องได้รับอนุญาตจาก Proxy จะระบุเฮดเดอร์
Request (1.1)	ให้ทราบซึ่งบราวเซอร์สามารถส่ง Requestใหม่โดยระบุเฮดเดอร์ Proxy-Authorization ด้วย
408 Request Timeout (1.1)	บราวเซอร์ไม่ส่ง Request ตามเวลาที่เซิร์ฟเวอร์รอได้
409 Conflict (1.1)	บราวเซอร์ส่งข้อมูลที่มีความหมายขัดแย้งกันเอง
410 Gone (1.1)	เอกสารที่ต้องการไม่ได้อยู่บนเซิร์ฟเวอร์แล้ว
411 Length Required (1.1)	เซิร์ฟเวอร์ต้องการให้ระบุ Content-Length ด้วย
412 Precondition Failed (1.1)	เงื่อนไขบางอย่างที่กำหนดใน Request Header ตกไป
413 Request Entity Too Large (1.1)	ข้อมูลที่ส่งมามีขนาดใหญ่เกินกว่าที่เซิร์ฟเวอร์จะรองรับได้
414 Request URI Too Long(1.1)	ค่า URL ที่ระบุ ยาวเกินไป
415 Unsupport Media Type(1.1)	ไม่รองรับการทำงานของ Media Type
500 Internal Server Error	เซิร์ฟเวอร์มีปัญหา
501 Not Implemented	เซิร์ฟเวอร์ไม่รองรับคำสั่งที่สั่งไป
502 Bad Gateway	Proxy Server รับคำสั่งที่ไม่ถูกต้องจากเว็บเซิร์ฟเวอร์
503 Service Unavailable	เซิร์ฟเวอร์กำลังทำงานอื่นอยู่ ไม่สามารถให้บริการได้ในขณะนี้
504 Gateway Timeout (1.1)	ในขณะทำหน้าที่เป็น Gateway หรือ Proxy ไม่ได้รับข้อมูลตอบจากเซิร์ฟเวอร์ปลายทางในเวลาที่กำหนด
505 HTTP Version not Supported (1.1)	เซิร์ฟเวอร์ไม่รองรับการทำงานของ HTTP เวอร์ชั่นนั้น ๆ

รูปที่ 2-16 รหัสสถานะ (ต่อ)

2.4.2.1.2 ส่วนเฮดเดอร์ย่อย

เฮดเดอร์ย่อยเป็นส่วนที่ใช้บอกรายละเอียดต่าง ๆ ของข้อมูล ทั้งการร้องขอและตอบสนอง โดยมีลักษณะเป็นข้อความธรรมดา ซึ่งมีรูปแบบการเขียนดังนี้

Header-name: Value

รายละเอียดปลีกย่อยของเฮดเดอร์ ได้แก่

- เฮดเดอร์อาจมีหลายประการ แต่ท้ายเฮดเอร์แต่ละรายการต้องปิดด้วยรหัสลงบรรทัดใหม่
- Header-name หรือชื่อของเฮดเดอร์จะพิมพ์ตัวเล็กหรือใหญ่ก็ได้ ไม่มีผลต่อการตีความหมาย
- หลังเครื่องหมาย : ของเฮดเดอร์แต่ละรายการอาจเป็นช่องว่าง (Space) หรือ แท็บ (Tab) ก็ได้
- เฮดเดอร์รายการใดที่ขึ้นต้นด้วยช่องว่างหรือแท็บ จะเสมือนว่าเป็นส่วนหนึ่งของเฮดเดอร์รายการก่อนหน้านี้ 1 บรรทัด

ใน HTTP เวอร์ชัน 1.0 กำหนดให้มีเฮดเดอร์ได้ถึง 16 รายการ แต่อาจจะไม่มีแม้แต่รายการเดียวเลยก็ได้ ส่วน HTTP เวอร์ชัน 1.1 กำหนดได้ 46 รายการ แต่ต้องมีเฮดเดอร์อย่างน้อย 1 รายการ คือ Host: เพื่อบอกชื่อโฮสต์และโดเมนเนม ในที่นี้จะขอกล่าวเพียงเฮดเดอร์ของ HTTP เวอร์ชัน 1.0 ซึ่งมีด้วยกันอยู่ 16 รายการ ดังแสดงในรูปที่ 2-10

เฮดเดอร์ย่อย	รายละเอียด
1. Allow	กำหนดเมธอดที่สนับสนุน จุดประสงค์ของข้อมูลนี้เพื่อเจาะจงเมธอดการร้องขอจากไคลเอนต์ โดยจะไม่สนับสนุนเมธอด POST
2. Authorization	สำหรับผู้ที่ต้องการการรับรอง (authentication) ด้วยตัวเองจากเซิร์ฟเวอร์ (ซึ่งอาจไม่จำเป็น) หลังจากได้รับการตอบสนองด้วยรหัสสถานะ 401 (Unauthorized) ควรจะมีการเพิ่มเฮดเดอร์นี้เข้าไปด้วย
3. Content-Encoding	ระบุการเข้ารหัสของข้อมูล
4. Content-Length	ใช้ระบุขนาดของข้อมูลในบล็อกข้อมูลมีหน่วยเป็นไบต์ (byte) เพื่อที่ผู้รับจะได้ทราบว่ามีข้อมูลส่งมาให้กี่ไบต์
5. Content-Type	ใช้ระบุชนิดของข้อมูลในบล็อกข้อมูลว่าเป็นข้อมูลประเภทไหน เช่น หากเป็นเอกสารแบบ HTML จะต้องระบุเป็น text/html ถ้าเป็นไฟล์รูปกราฟิกแบบ gif ก็ต้องระบุเป็น image/gif เป็นต้น แต่สำหรับเว็บบราวเซอร์รุ่นใหม่ ๆ แล้ว หากข้อมูลที่ได้รับไม่มีการระบุว่าเป็นประเภทไหนแล้ว เว็บบราวเซอร์จะถือว่าเป็นประเภท text/html เสมอ
6. Date	ข้อมูลส่วนนี้ใช้ระบุวันเวลาที่ข้อมูลการร้องขอถูกส่งมา
7. Expires	เป็นเฮดเดอร์ในส่วนการตอบสนองของเซิร์ฟเวอร์ใช้กำหนดวันที่หมดอายุของไฟล์ที่ส่งไปให้ไคลเอนต์ รายการนี้สามารถใช้ในทางเทคนิคเพื่อป้อนการเก็บไฟล์ไว้ในแคช (Cache) จากเว็บบราวเซอร์อย่าง Navigator ได้ โดยระบุวันที่ใน Expires: ให้ย้อนจากวันเวลาปัจจุบันนาน ๆ เมื่อเว็บบราวเซอร์รับไฟล์ไปก็จะเข้าใจว่าไฟล์นี้หมดอายุแล้ว ถึงแม้จะนำเนื้อหาไปแสดงในวินโดว์เว็บบราวเซอร์แต่จะไม่เก็บไว้ในแคช ทำให้ทางเซิร์ฟเวอร์มั่นใจได้ว่า ทุกครั้งที่ไคลเอนต์ร้องขอไฟล์จะต้องวิ่งมาขอจากเซิร์ฟเวอร์ใหม่ทุกครั้ง ถึงแม้จะเป็นการร้องขอไฟล์เดิม ๆ และทางเซิร์ฟเวอร์ไม่มีการอัพเดตก็ตาม
8. From	เป็นเฮดเดอร์ในส่วนการร้องขอของไคลเอนต์ จะประกอบด้วย e-mail address ของผู้ใช้ที่ควบคุมการส่งข้อมูลการร้องขอ

รูปที่ 2-17 รายละเอียดของเฮดเดอร์ย่อยของโพรโตคอล HTTP

เฮดเดอร์ย่อย	รายละเอียด
9. If-Modified-Since	เป็นเฮดเดอร์ในส่วนการร้องขอของไคลเอนต์ ถูกใช้ร่วมกับเมธอด GET เพื่อใช้ในการกำหนดเงื่อนไขบอกแก่เซิร์ฟเวอร์ว่า ถ้าไฟล์ที่ร้องขอไปมีการแก้ไขหลังจากวันที่ได้ระบุในเฮดเดอร์นี้ เซิร์ฟเวอร์จึงค่อยส่งไฟล์นั้นมาให้ แต่ถ้ายังไม่ได้มีการแก้ไขหลังช่วงวันที่ระบุ เซิร์ฟเวอร์ไม่ต้องส่งไฟล์นั้นมาให้ เพียงแต่เซิร์ฟเวอร์รหัสสถานะตอบสนองมาเป็น 304 (not Modified)แทน
10. Last-Modified	ข้อมูลนี้ใช้ระบุวันเวลาครั้งล่าสุดที่มีการ modify ข้อมูลนั้น ซึ่งจะต้องบอกวันที่และเวลาในรูปแบบของเวลามาตรฐาน GMT
11. Location	เป็นเฮดเดอร์ส่วนการตอบสนองของเซิร์ฟเวอร์ที่ใช้แจ้งแหล่งที่อยู่ของข้อมูลที่ไคลเอนต์ต้องการ สำหรับในรหัสสถานะการตอบสนองที่ 3xx Location จะชี้ URL ที่ใช้สำหรับรีไดเรก (redirect)
12. Pragma	ข้อมูลในส่วน Pragma นี่ถูกใช้รวมกับ implementation-specific derictives ที่สามารถนำมาใช้กับผู้รับตลอดสายการร้องขอ/ตอบสนอง โดยข้อมูล prama directoves ทั้งหมดจะกำหนดการกระทำจากในมุมมองของโพรโตคอล
13. Referer	เป็นเฮดเดอร์ในส่วนของการร้องขอของไคลเอนต์ เป็น URI ของแหล่งที่มาของการร้องขอ ข้อมูลส่วนนี้จะทำให้เซิร์ฟเวอร์สร้างรายการการย้อนหลัง (lists of back-links) การล็อกอินเข้าระบบ การจัดการ Cache และอื่น ๆ มันยังส่งผลให้ ข้อมูลในส่วน Referer นี้จะต้องไม่ถูกส่งมาถ้าไม่ได้มาจากแหล่งที่มี URI ของตัวเอง ตัวอย่างเช่น จาก คีย์บอร์ดของผู้ใช้เอง
14. Server	เป็นเฮดเดอร์ในส่วนการตอบสนองจากเซิร์ฟเวอร์ ประกอบด้วยข้อมูลเกี่ยวกับซอฟต์แวร์ซึ่งทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ โดยมีรูปแบบคือ Program-name/x.xx

รูปที่ 2-18 รายละเอียดของเฮดเดอร์ย่อยของโพรโตคอล HTTP (ต่อ)

2.4.2.2 ข้อมูลที่ต้องการรับ-ส่ง

จากรูปที่ 2-3 ในส่วนสุดท้าย ซึ่งต่อจากส่วนของเฮดเดอร์ย่อยของเฮดเดอร์ HTTP จะเป็นส่วนของบล็อกข้อมูล ซึ่งเป็นส่วนของข้อมูลที่เราต้องการส่งจริง อาจจะเป็น HTML file หรือ Text file หรือข้อมูลชนิดอื่นๆ

2.5 ระบบตรวจจับการบุกรุกทางเครือข่าย Network Intrusion Detection System (NIDS)

ระบบตรวจจับการบุกรุก Intrusion Detection System (IDS) เป็นระบบที่จะคอยมองดูข้อมูลที่เป็นไปในลักษณะของการบุกรุก โดยอาศัยการกำหนดรูปแบบของลักษณะการทำงานหรือรูปแบบของข้อมูล แล้วตัดสินว่าเป็น รูปแบบข้อมูลที่ถูกต้อง , ผิดปกติ หรือน่าสงสัยหรือไม่ โดยจะแบ่งออกเป็น 2 ลักษณะคือ Network Based และ Host Based

ระบบตรวจจับการบุกรุกทางเครือข่าย (NIDS) ทำหน้าที่ตรวจสอบแพ็กเก็ตที่อยู่บนเครือข่าย โดยทั่วไปแล้วจะมีการตั้งให้ทำงานเป็น promiscuous โหมด โดยจะคอยตรวจดูและวิเคราะห์ข้อมูลทั้งหมดที่ผ่านเข้ามาปีนแบบ real-time โดยในขั้นแรกนั้นจะตรวจดูก่อนว่าเป็นข้อมูลที่ผ่านเข้ามาควรจะส่งต่อเข้าไปตรวจสอบในโมดูลที่ใช้ตรวจดูการโจมตีหรือไม่ ซึ่งในขั้นนี้จะช่วยในการเพิ่มประสิทธิภาพและความแม่นยำในการทำงานของระบบตรวจจับการบุกรุกทางเครือข่ายด้วย โดยจะอนุญาตให้ข้อมูลที่รู้ว่าเป็นแพ็กเก็ตที่ไม่ประสงค์ร้าย ไม่ถูกนำเข้าไปตรวจสอบและถูกกรองออกไป ตัวอย่างเช่น เกิดเหตุการณ์ที่น่าสงสัยคือ มีข้อมูล SNMP อยู่ในระบบเครือข่าย แต่เราทราบว่าข้อมูลเหล่านี้ถูกสร้างโดยเครื่องในระบบเครื่องหนึ่ง เราก็สามารถที่จะกรอง SNMP โดยไม่ต้องนำเข้ามาตรวจสอบได้ ดังนั้นการปรับแต่งจึงเป็นเรื่องสำคัญ กรณีที่ตั้งค่าการใช้งานผิดพลาดอาจจะเป็นสาเหตุให้เครื่องที่ทำหน้าที่ NIDS ต้องทำงานมากกว่าที่คาดคิดไว้

สำหรับโมดูลที่ใช้ในการตรวจดูการโจมตีมีลักษณะดังนี้คือ pattern or signature , frequency และ anomaly detection ในการตรวจจับการโจมตีโดยวิธีการ pattern matching นี้มีข้อดีคือ มีความแม่นยำสูงและสามารถเพิ่มรูปแบบใหม่ ๆ ที่ทันสมัยเข้าไปได้เรื่อย ๆ แต่ข้อเสียของวิธีนี้คือ สามารถเกิด False Negative ได้ คือตรวจจับได้ข้อมูลปกติที่มีข้อมูลภายในคล้ายการบุกรุก หรือในกรณีที่ผู้บุกรุกเปลี่ยนแปลงข้อมูลภายในเล็กน้อยก็สามารถที่จะผ่านการตรวจจับได้ ลักษณะนี้เรียกว่า False Positive

เมื่อสามารถตรวจจับการบุกรุกได้ โมดูลที่ใช้ในการรายงานผลก็จะทำหน้าที่โดยจะนำเสนอออกมาในหลายรูปแบบเช่นแจ้งเตือน , เก็บลง log หรือตอบสนองในรูปแบบอื่น ๆ ซึ่งตัว NIDS บางตัวสามารถที่จะทำงานร่วมกับไฟร์วอลล์ โดยอาจจะสั่งให้ไฟร์วอลล์ปิดกั้นการสื่อสารของไอพีแอดเดรสที่พบว่าเป็นการโจมตีเข้ามาได้

ข้อดีของระบบตรวจจับการบุกรุกทางเครือข่าย

1. ค่าใช้จ่ายต่ำ เนื่องจาก NIDS จะถูกนำไปติดตั้งยังจุดที่มีข้อมูลไหลผ่านในปริมาณมากเพื่อให้สามารถดูแลระบบได้จำนวนมากที่สุด โดยที่มีความจำเป็นที่จะต้องติดตั้งโปรแกรมลงไปในเครื่องโฮสต์ต่าง ๆ ซึ่งจุดนี้ทำให้ค่าใช้จ่ายในการดูแลและการจัดการต่ำ โดยที่ได้ประสิทธิภาพสูง
2. การวิเคราะห์แพ็กเก็ต NIDS จะตรวจดูเฮดเดอร์ (Header) ของแพ็กเก็ตว่าเป็นแพ็กเก็ตที่ผิดปกติหรือน่าสงสัยหรือไม่ ปัจจุบัน การ Denial of Service (DoS) โดยใช้ไอพีนั้นสามารถตรวจจับได้โดยดูจากเฮดเดอร์ของแพ็กเก็ต ตัวอย่างเช่นการโจมตีแบบ LAND ซึ่งจะกำหนดทั้งผู้ส่งและผู้รับเป็นไอพีเดียวกัน แพ็กเก็ตที่ถูกสร้างโดยวิธีนี้ จะทำให้เครื่องปลายทางสร้างการเชื่อมต่อกับเครื่องของตนเองและจะทำให้เครื่องปลายทางทำงานช้าลงหรือใช้งานไม่ได้เลย ซึ่งการโจมตีลักษณะนี้สามารถที่จะตรวจจับได้โดยง่ายและรวดเร็วและการโจมตีที่ใช้การแฟร็กเมนต์ของแพ็กเก็ตเช่น TearDrop ก็สามารถที่จะตรวจจับได้เช่นกัน นอกเหนือจากการดูในส่วนของเฮดเดอร์แล้วยังสามารถที่จะตรวจดูเนื้อหาภายในของข้อมูลเพื่อที่จะใช้ตรวจหาว่ามีคำสั่งหรือข้อความที่เข้าข่ายการโจมตีหรือไม่
3. การตรวจจับและตอบสนองแบบ real-time เพื่อ NIDS สามารถตรวจจับการบุกรุกหรือแพ็กเก็ตที่น่าสงสัย ก็จะสามารถรายงานผลหรือแจ้งเตือนออกมาได้อย่างรวดเร็วและทันท่วงที ตัวอย่างเช่นเมื่อผู้บุกรุกโจมตีโดยใช้วิธี Dos โดยผ่าน TCP NIDS ก็จะส่ง TCP Reset ออกไปเพื่อจะหยุดการโจมตีก่อนที่ระบบจะใช้การไม่ได้ การตอบสนองแบบ real-time นี้จะช่วยให้เราสามารถแก้ปัญหาได้รวดเร็วมากยิ่งขึ้น หรือทำให้ปัญหาเบาบางลงได้
4. การลบหลักฐานของผู้บุกรุกนั้นทำได้ยาก เนื่องจากการทำงานของ NIDS นั้นจะทำงานและรายงานผลแบบ real-time ทันทีที่พบว่ามีการบุกรุกเกิดขึ้น ข้อมูลของผู้บุกรุกจะถูกแจ้งออกมาทันที และเราสามารถนำหลักฐานหรือข้อมูลเหล่านี้ไปค้นหาสาเหตุหรือเบาะแสของผู้บุกรุกได้อีกด้วย
5. สามารถใช้ NIDS ในการดูแลนโยบายทางด้านความปลอดภัยในแง่อื่น ๆ ได้เช่นกรณีที่มีการใช้งานการเข้ารหัส ถึงแม้ว่า NIDS นั้นไม่สามารถที่จะอ่านและเข้าใจข้อมูลที่เข้ารหัสนั้น ๆ แต่ก็สามารถที่จะนำไปตรวจจับข้อมูลที่ไม่มีการเข้ารหัสได้ ซึ่งไม่ควรที่จะมีอยู่ในระบบนี้
6. การทำงานของ NIDS นั้นไม่ขึ้นอยู่กับระบบปฏิบัติการ ไม่ต้องคำนึงว่าโฮสต์ที่ต้องดูแลเป็นระบบปฏิบัติการอะไร และไม่ต้องติดตั้งโปรแกรมลงไปที่โฮสต์นั้น ๆ

2.6 กฎการคำนวณแบบบอยเออร์มัวร์ (Boyer Moore Algorithm)

เป็นวิธีการทำการเปรียบเทียบสตริง (string matching) ที่มีประสิทธิภาพมาก โดยจะมองเป็นสายข้อมูล (String) และคำที่เราต้องการค้นหา (Pattern)

วิธีการคือ scan ตัวอักษรใน pattern จากซ้ายไปขวาเพื่อเทียบกับ string โดยดูว่าตัวอักษรสุดท้ายของ pattern ตรงกับอักษรใดใน string และต้อง shift ตัว pattern ออกไปจำนวนกี่ครั้งเมื่อนำค่าตัวอักษรใน string นั้นไปเทียบโดยฟังก์ชั่นที่ได้มีการคำณวนไว้ก่อนแล้วเพื่อให้เลื่อนตัวอักษรใน pattern กับ string นั้นให้ตรงกันจากนั้นก็ให้ตรวจสอบตั้งแต่ตำแหน่งสุดท้ายของ pattern ว่ามีค่าเท่ากันตลอดทั้ง pattern หรือไม่ ถ้าใช่ แสดงว่ามี pattern นั้นอยู่ใน string แต่ถ้าไม่ตรงก็ให้ shift ออกไปอีกเท่ากับความยาวของ pattern นั้น แล้วกลับไปค้นหาตั้งแต่ต้นใหม่

ตัวอย่างการทำงานของ Boyer-Moore Algorithm

Pattern	E	X	A	M	P	L	E
ค่าที่ใช้ในการ Shift	6	5	4	3	2	1	0

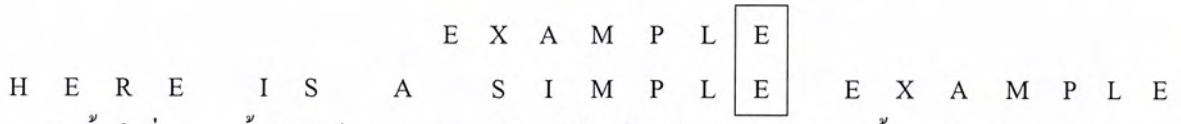
String = HERE IS A SIMPLE EXAMPLE

```
E X A M P L E
H E R E   I S   A   S I M P L E   E X A M P L E
```

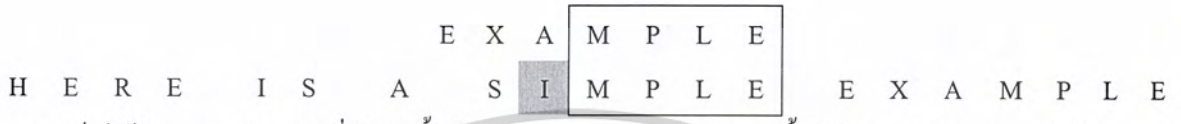
ขั้นตอนแรกจะพิจารณาตัวตำแหน่งสุดท้ายของ pattern ก่อนว่าตรงกับตำแหน่งใดใน string ซึ่งในที่นี้คือ S เมื่อนำ S ไปตรวจสอบการ shift ในตารางที่ได้คำนวณไว้ก่อนหน้านี้แล้วก็พบว่าต้องเลื่อนไปเท่ากับความยาวของ pattern นั้นซึ่งในที่นี้คือ 7

```
          E X A M P L E
H E R E   I S   A   S I M P L E   E X A M P L E
```

เมื่อพิจารณาตำแหน่งท้ายสุดของ pattern ซึ่งก็คือ E ตรงกับอักษร P ใน string เราจึงนำค่าของ P ไปตรวจสอบในตารางว่าจะต้องเลื่อนไปจำนวนกี่ตำแหน่งโดย ค่าที่ได้คือ 2 จึงต้องเลื่อนไปอีกสองตำแหน่ง



จากนั้นจึงเริ่ม scan ตั้งแต่ตัวท้ายของ pattern คือ E ใส่ไปด้านซ้ายว่าเท่ากันตลอดทั้งความยาวของ pattern หรือไม่



จะเห็นได้ว่า จะตรงกันเพียง สี่ตัวเท่านั้นคือ MPLE แต่ว่าตัว I ใน string นั้นไม่ตรงกันกับ pattern จึงต้องเลื่อนไปอีกเท่ากับความยาวของ pattern

ตำแหน่งของ string ที่ตรงกันกับตำแหน่งสุดท้ายของ pattern คือ L ซึ่งนำไปเทียบในตารางเพื่อตรวจดูว่าต้องเลื่อนจำนวนกี่ครั้ง ซึ่งในที่นี้ได้ค่าออกมาเท่ากับ 1

จากนั้นก็เริ่มเปรียบเทียบตั้งตำแหน่งสุดท้ายของ pattern

เมื่อเปรียบเทียบจนถึงตำแหน่งแรกของ pattern ก็พบว่าเท่ากันจึงให้ค่าว่าพบ pattern นี้ใน string

2.7 การโจมตีเว็บเซิร์ฟเวอร์ (Web Server Intrusion)

ลักษณะวิธีที่ใช้ในการโจมตีเว็บเซิร์ฟส่วนใหญ่เป็นผลอันเนื่องมาจากช่องโหว่ที่มีอยู่ในตัวโปรแกรมเอง อันจะทำให้ผู้บุกรุกอาศัยจุดนี้ในการโจมตีเว็บเซิร์ฟเวอร์โดยวิธีต่าง ๆ สามารถจัดลักษณะผิดพลาดของโปรแกรม อันเป็นผลให้สามารถโจมตีได้เป็น 4 วิธีดังนี้คือ

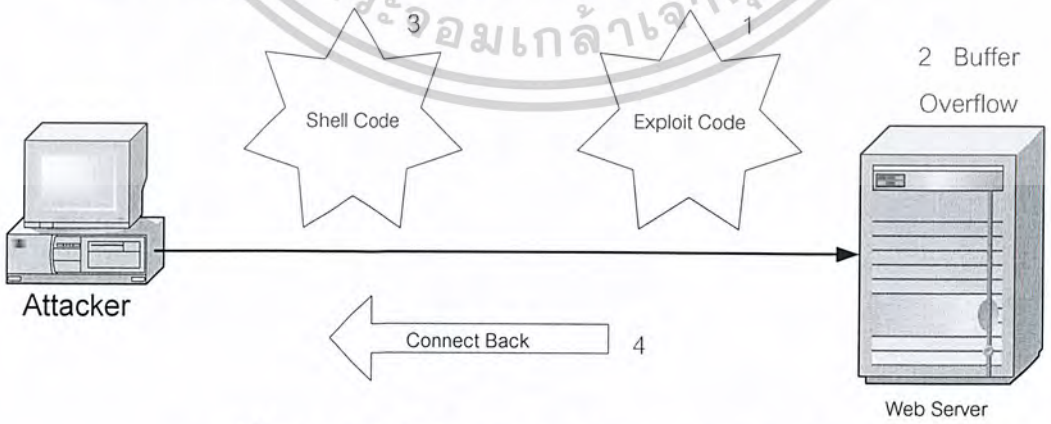
1. Input Validation Error ผู้บุกรุกจะอาศัยข้อผิดพลาดของเว็บเซิร์ฟเวอร์กรณีที่มีการแปลงข้อมูลหรือรับข้อมูลเข้าไปผิดรูปแบบ ตัวอย่างของข้อผิดพลาดประเภทนี้ได้แก่ เว็บเซิร์ฟเวอร์ Apache ที่ถูกป้อนข้อมูลที่เป็น / (slash) เข้าไปเป็นจำนวนมาก (อย่างน้อย 4000 ตัว ขึ้นอยู่กับชนิดของระบบปฏิบัติการ) จะสามารถทำ Directory Traversal ได้ โดยมีลักษณะการป้อนข้อมูลเข้าไปในลักษณะนี้

GET /cgi-bin/// HTTP/1.0

และข้อผิดพลาดของเว็บเซิร์ฟเวอร์ IIS ของ Microsoft Windows 2000 ที่มีข้อผิดพลาดในการแปลงข้อมูลชนิด Unicode ก็จะเป็นสาเหตุให้ผู้บุกรุกสามารถทำ Directory Traversal ได้เช่นเดียวกัน โดยข้อมูลที่ป้อนมีลักษณะดังนี้

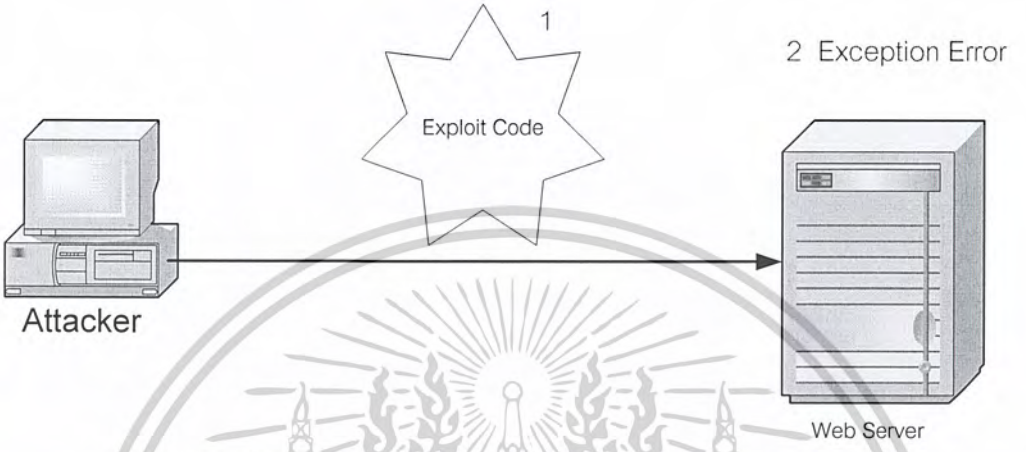
GET /scripts/..%c0%af../winnt/system/cmd.exe+/c+dir:\ HTTP/1.0

2. Boundary Condition Error ข้อผิดพลาดนี้เกิดมาจากการจองพื้นที่ในหน่วยความจำเอาไว้ไม่เพียงพอกับขนาดของข้อมูลที่ป้อนเข้าไป หรือเกิด buffer overflow เป็นผลให้ผู้บุกรุกสามารถที่จะเข้าไปเขียนข้อมูลในตำแหน่งหน่วยความจำนั้น ๆ และสั่งให้ทำงานได้



รูปที่ 2-19 การเกิดข้อผิดพลาดจาก Boundary Condition Error

3. Failure to Handle Exceptional Condition ความผิดพลาดของตัวโปรแกรมชนิดนี้เป็นผลอันเนื่องมาจากการที่ตัวโปรแกรมไม่สามารถที่จะตัดสินใจได้ว่า ควรจะปฏิบัติงานอย่างไรกับข้อมูลที่ได้มา เป็นผลให้โปรแกรมอาจจะหลุดออกไปอยู่ในสถานะที่ไม่สามารถทำงานได้ ผู้บุกรุกจึงนำข้อผิดพลาดนี้มาใช้ในการทำ Denial of Service (DoS)



รูปที่ 2-20 การเกิดข้อผิดพลาดจาก Failure to Handle Exceptional Condition

4. Architecture Design Error ความผิดพลาดนี้เป็นผลมาจากการออกแบบที่ผิดพลาดมีผลทำให้ผู้บุกรุกได้รับข่าวสารหรือข้อมูลที่ควรจะเป็นความลับหรือไม่น่าจะเปิดเผยออกไป ซึ่งผู้บุกรุกอาจจะอาศัยข้อมูลเหล่านี้ไปใช้ประโยชน์ในการโจมตีรูปแบบอื่นได้ เช่นกรณีของ Apache ในรุ่นเก่า ๆ ซึ่งจะให้ข้อมูลเกี่ยวกับหมายเลข inode ภายในเครื่องไปกับ Header Etag ของ HTTP โปรโตคอล หรือ Apache 2.0 ที่เปิดเผยพาธของ CGI ว่าอยู่ที่ตำแหน่งใดภายในเครื่อง

2.8 โปรเซส (Process)

เวลาที่จะเขียนซอร์สโปรแกรม (source code) ภาษาซี โปรแกรมเมอร์สร้างไฟล์นั้นบนดิสก์ที่มีประโยคภาษาซีที่ทำงานตามต้องการ โดยในแต่ละโปรแกรมก็จะมีรายละเอียดต่างๆกัน ซึ่งโดยปกติแล้วนั้นจะมีนามสกุลเป็น .c

ต่อมาคอมไพเลอร์ภาษาซีจะแปลซอร์สภาษาซีนั้นไปเป็นออบเจ็คไฟล์ (object file) แล้วลิงค์ออบเจ็คไฟล์แต่ละอันเข้าด้วยกันเป็นโมดูล (module) ที่สามารถเอ็กซ์ซีคิว (execute)ได้ โดยโปรแกรมก็คือไฟล์ที่เก็บโมดูลที่เอ็กซ์ซีคิวได้เหล่านั้นนั่นเอง เมื่อโปรแกรมถูกสั่งให้ทำงาน (run) ระบบปฏิบัติการจะทำสำเนาโมดูลเหล่านั้นไปยังหน่วยความจำหลัก (main memory) ในรูปของโปรแกรมอิมเมจ(image program) โดยโปรเซส ก็คือ โปรแกรมที่กำลังทำงานอยู่ในหน่วยความจำหลังนั่นเอง ซึ่งแต่ละโปรเซสจะมีแอดเดรส (address) และสถานะ(status) ของตนเอง

เวลาที่โปรแกรมจะกลายเป็นโปรเซสก็คือ เมื่อระบบปฏิบัติการอ่านโปรแกรมเข้าหน่วยความจำแต่การจอง (allocation) หน่วยความจำให้กับโปรแกรมนั้นยังไม่ถือว่าเป็นการสร้างโปรเซสจะต้องมี ID (Process ID) ซึ่งระบบปฏิบัติการใช้แยกความแตกต่างของแต่โปรเซส

กล่าวคือเมื่อระบบปฏิบัติการเพิ่มเติมข้อมูลของโปรแกรมในส่วนเคอร์เนลดาต้า (Kernel data) และจัดสรรทรัพยากรของระบบให้กับโปรแกรมโค้ดแล้วโปรแกรมจะกลายเป็นโปรเซส

2.8.1 Process ID

Unix จะระบุโปรเซสโดยให้เลขที่ไม่ซ้ำเรียกว่า Process ID โปรเซสที่สร้างโปรเซสใหม่เรียกว่า พาเรนท์ (parent) ของโปรเซสที่ถูกสร้างซึ่งจะเรียกว่า ไชลด์ (child) ของโปรเซสที่เป็น พาเรนท์

```
#include <sys/type.h>
#include <unistd.h>

pid_t getpid (void)
```

แต่ละโปรเซสจะมียูเซอร์เป็นเจ้าของ (owner) หากเจ้าของโปรเซสนั้นมีสิทธิตามนั้น โดยแต่ละยูสเซอร์จะมีหมายเลขประจำตัวเรียกว่า user ID โปรเซสจะรู้หมายเลขของเจ้าของโปรเซสโดยใช้คำสั่ง getuid โดยโปรเซสจะมีสิทธิตามค่าของ effective user ID ซึ่ง euid นี้จะเปลี่ยนแปลงได้ระหว่างเอ็กซ์ซีคิว

```
#include <sys/type.h>
#include <unistd.h>
uid_t getuid (void)
uid_t geteuid (void)
```

ตัวอย่าง โปรแกรมจะพิมพ์ process ID , parent process ID ของมันและ user ID ของเจ้าของ

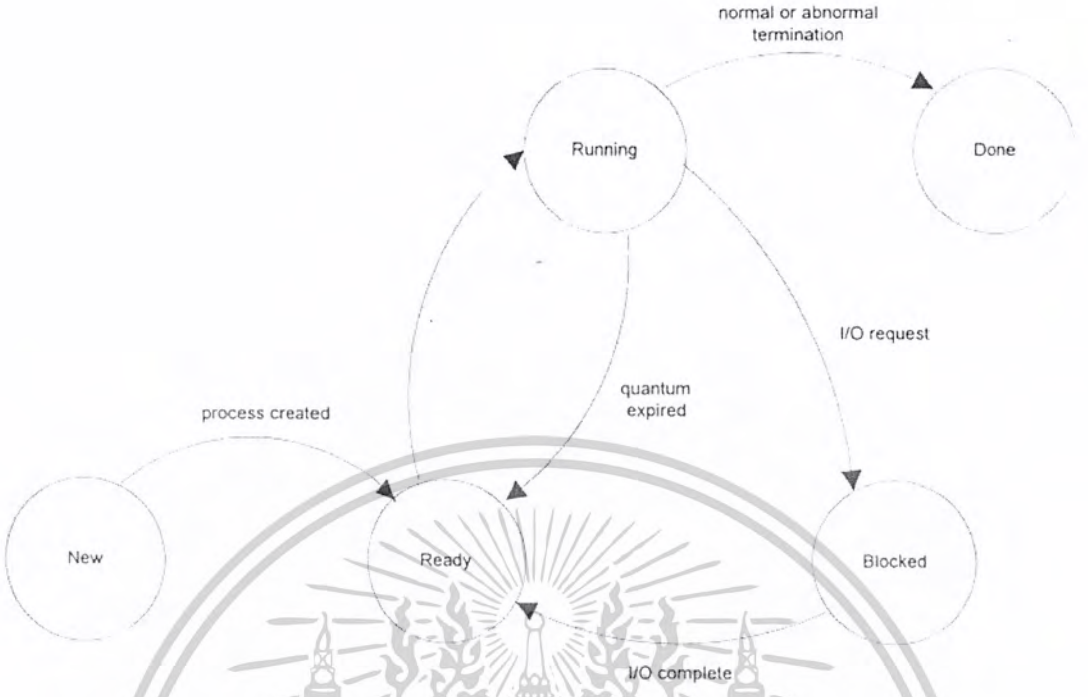
```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

void main (void)
{
        printf("Process ID : %1d\n",(long)getpid());
        printf("Parent Process ID : %1d\n",(long)getppid());
        printf("Parent Process ID : %1d\n",(long)getuid());
}
```

2.8.2 Process state

โปรเซสจะมีสถานะเปลี่ยนแปลงตามเวลาโดยจะขึ้นกับกิจกรรมที่โปรเซสกำลังทำอยู่ โดยระบบปฏิบัติการส่วนใหญ่จะมีสถานะของโปรเซสดังตารางและในรูปไดอะแกรมจะแสดงการเปลี่ยนแปลงของโปรเซสจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง

สถานะ	ความหมาย
New	โปรเซสกำลังถูกสร้างขึ้น
Running	คำสั่งของโปรเซสกำลังถูกรัน
Blocked	โปรเซสกำลังคอยอีฟเวนต์ ตัวอย่างเช่น I/O
Ready	โปรเซสกำลังคอยที่จะถูกส่งเข้าไปรัน
Done	โปรเซสเสร็จงานและคืนทรัพยากรให้กับระบบ

ตารางที่ 2-1 แสดงสถานะต่างๆของโปรเซสและความหมาย



รูปที่ 2-21 แสดงสถานะต่างๆของโปรเซส

เมื่อโปรแกรมกำลังกลายเป็นโปรเซสกล่าวได้ว่ามันกำลังอยู่ในสถานะ new เมื่อการเปลี่ยนแปลงดังกล่าวเสร็จสิ้นโปรเซสจะถูกนำไปใส่คิวเพื่อคอยรันเรียกว่าสถานะ ready เมื่อโปรเซสถูกจัดให้รันโปรเซสก็จะอยู่ในสถานะ running ซึ่งโปรเซสจะถูกเอ็กซ์ซีคิวในซีพียู

โปรเซสจะอยู่ในสถานะ blocked เมื่อโปรเซสคอยอีฟเวนท์ (event) ซึ่งเรียกอีกอย่างหนึ่งว่าโปรเซสกำลังหลับอยู่ (sleep) หรืออีกนัยหนึ่งโปรเซสจะเข้าสู่ สถานะ blocked เมื่อโปรเซสทำการร้องขอ I/O

การที่มีการหยุดโปรเซสหนึ่งซึ่งกำลังอยู่แล้วเปลี่ยนเป็นอีกโปรเซสหนึ่งว่าการทำ คอนเท็กซ์สวิทช์ (context switch) คอนเท็กซ์ของโปรเซสได้แก่ข้อมูล (information) ของโปรเซสและเอ็นวิรอนเมนต์ (environment) ของโปรเซสพื่อที่โปรเซสจะกลับมาทำงานอีกครั้งหลังการ สวิทช์ ซึ่ง ส่วนเอ็กซ์ซีคิวเทเบิล (executable) สแต็ก (stack) รีจิสเตอร์ (register) และโปรแกรมเคาน์เตอร์ (program counter) ก็เป็นส่วนประกอบของคอนเท็กซ์

2.8.3 การสร้างโปรเซส และการฟอร์ก (fork) ของ Unix

ในระบบ Unix การสร้างโปรเซสทำได้โดยเรียกคำสั่ง fork โดยก็อปปี้เมมโมรีอิมเมจ (memory image) ของพาเรนท์ ซึ่งทั้งสองโปรเซสจะยังคงเอ็กซ์ซีคิว ต่อไปหลักจากทำคำสั่ง fork แล้ว

```
#include <sys/type.h>
#include <unistd.h>

pid_t fork(void)
```

fork จะรีเทิร์นค่า 0 แก่โปรเซสไชลด์ และรีเทิร์น process ID ของโปรเซสไชลด์แก่พาเรนท์ซึ่งจากจุดนี้เราสามารถทำให้ไชลด์และพาเรนท์ทำงานแยกจากกันได้

ตัวอย่าง หลังจากทำการ fork แล้ว ไชลด์ และ พาเรนท์ จะแสดง process ID ของตัวเอง

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

if((childpid = fork()) == 0 )
{
        fprintf (stderr,"I am the child , ID = %1d\n" , (long)getpid());
        // child code goes here
}
else if (childpid > 0 )
{
        fprintf (stderr, "I am the parent , ID = %1d\n" , (long)getpid());
        //parent code goes here
}
```

จากตัวอย่างโปรเซสเดิมจะได้ค่า childpid เป็นค่าของ process ID ของโปรเซสไชลด์มีค่าที่ไม่เท่ากับศูนย์และจะทำคำสั่ง fprintf อันที่สอง ส่วนโปรเซสไชลด์จะมีค่า childpid เท่ากับศูนย์และจะทำ fprintf อันแรกโดยว่าเอาท์พุทอาจจะเรียงลำดับกันอย่างไรก็ได้เนื่องจากโปรเซสทั้งสองนั้นทำงานพร้อมกัน

ตัวอย่าง การสร้างโปรเซสแบบลูกโซ่ (chain process)

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int I;
int n;
pid_t childpid;
for (I=1;I<n;I++)
        if(childpid = fork())
                break;
fprintf (stderr,"This is process %1d with parent %1d\n", (long)getpid() , (long)getpid());
```

ในการเรียกคำสั่ง fork แต่ละครั้งพาเรนท์จะได้ childpid ที่ไม่เป็นศูนย์และจะออกจากลูป ส่วนของไชลด์จะมีค่าเป็นศูนย์และจะเป็นพาเรนท์ในลูปต่อไป

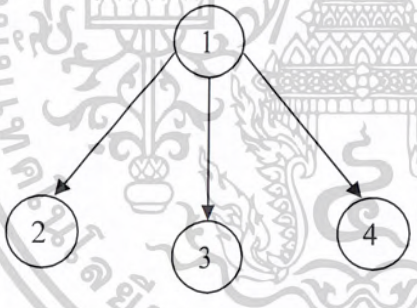
ถ้า n = 4 จะแสดงได้ดังรูป โปรเซสไชลด์จะ fork ต่อไปเรื่อยๆ

รูปที่ 2-22 แสดงการ fork โปรเซส

ตัวอย่าง การสร้างโปรเซสแบบใบพัด (fan of process)

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int I;
int n;
pid_t childpid;

for (I=1;I<n;++I)
        if((childpid = fork ()) <= 0)
                break;

fprintf (stderr,"This is process %1d with parent %1d\n",(long)getpid(),(long)getpid());
```

จากรูปจะแสดงการสร้างโปรเซสโปรเซสโดยพาเรนท์ จะสร้างโปรเซสไชลด์เป็นจำนวน n-1 โปรเซส



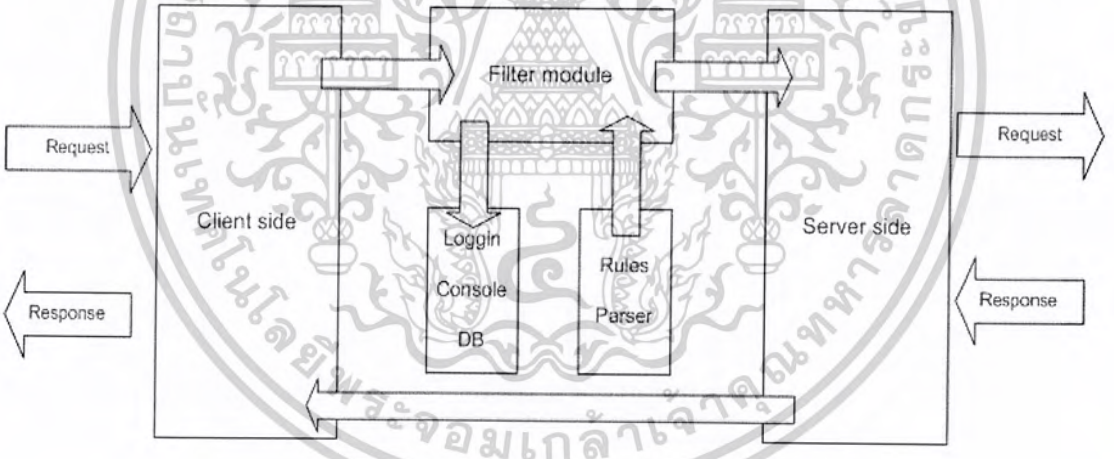
รูปที่ 2-23 แสดงการสร้างโปรเซสเป็นจำนวน n-1

บทที่ 3
การคำนวณ สร้างและการออกแบบ

3.1 เป้าหมายโครงงาน

ระบบนี้เป็นระบบที่มุ่งเน้นเพิ่มความปลอดภัยให้กับเว็บเซิร์ฟเวอร์ เป็นหลัก ซึ่งจะช่วยป้องกันการโจมตีที่ผ่านเข้ามาทางโปรโตคอลเอชทีทีพี (HTTP protocol) ซึ่งเมื่อมีการโจมตีเข้ามานั้น จะมีการบันทึกเก็บรายละเอียดของการโจมตีกับผู้ดูแลระบบได้หลายทางเพื่อความสะดวก ทั้งทางหน้าจอ ทางล็อกไฟล์ (log file) และฐานข้อมูล ระบบนี้จะทำงานในลักษณะทรานซ์พาเรนท์ (transparent) คือป้องกันไม่ให้ผู้โจมตีรู้ได้ว่ามีระบบนี้ทำงานอยู่

3.2 โครงสร้างของระบบโดยรวม



รูปที่ 3-1 แสดงการทำงานของระบบโดยรวม

ระบบนี้ได้แบ่งออกเป็นส่วนต่างหลายส่วนดังนี้

1 ส่วนฝั่งผู้ใช้บริการ (client side)

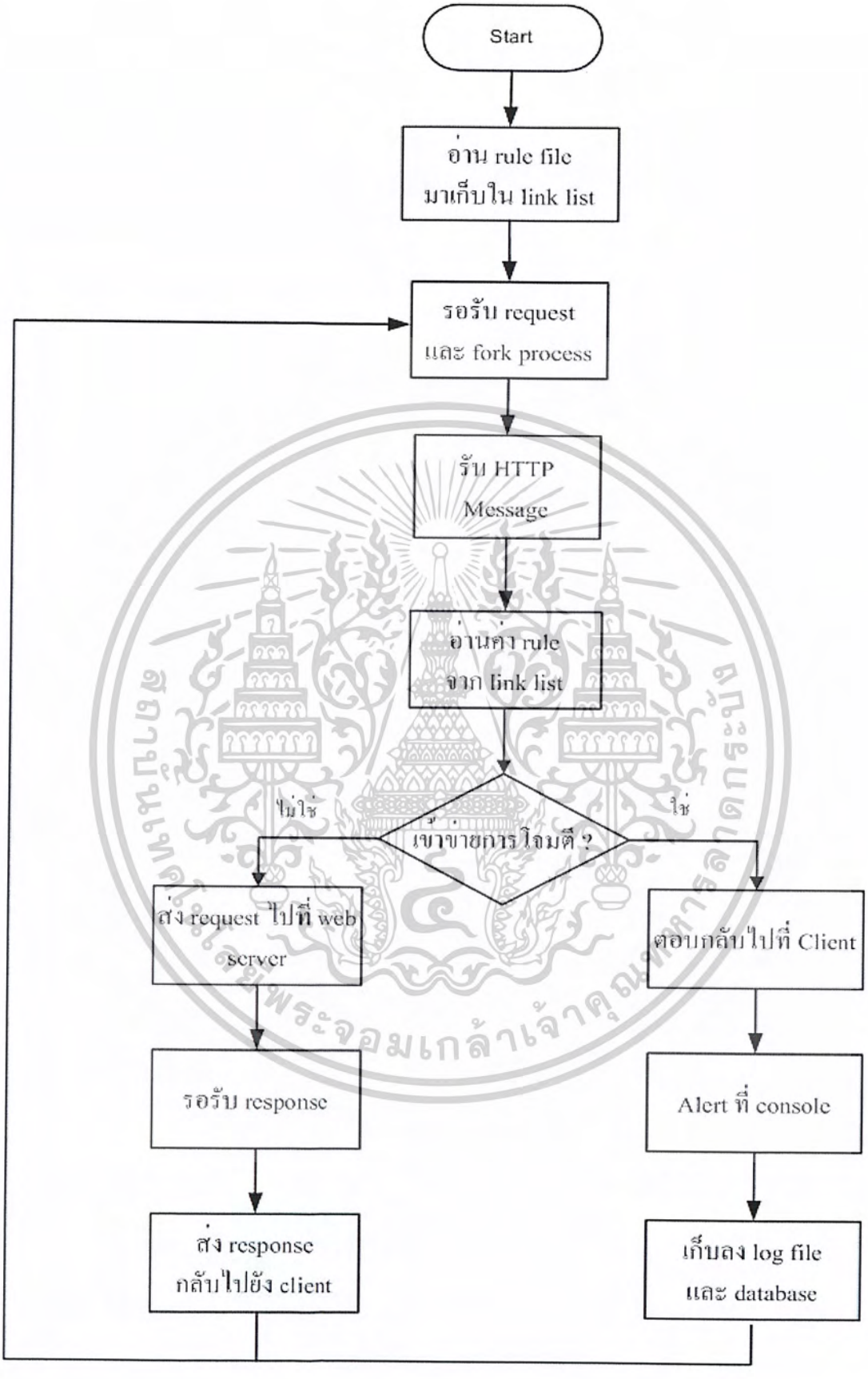
2 ส่วนกรองข้อมูล (filter side)

3 ส่วนฝั่งผู้ให้บริการ (server side)

4 ส่วนแจ้งเตือนความผิดปกติ (report module)

5 ส่วนอัพเดทและโปรแกรมแปลง (update and parser program)

6 ส่วนการปรับแต่ง (configuration module)

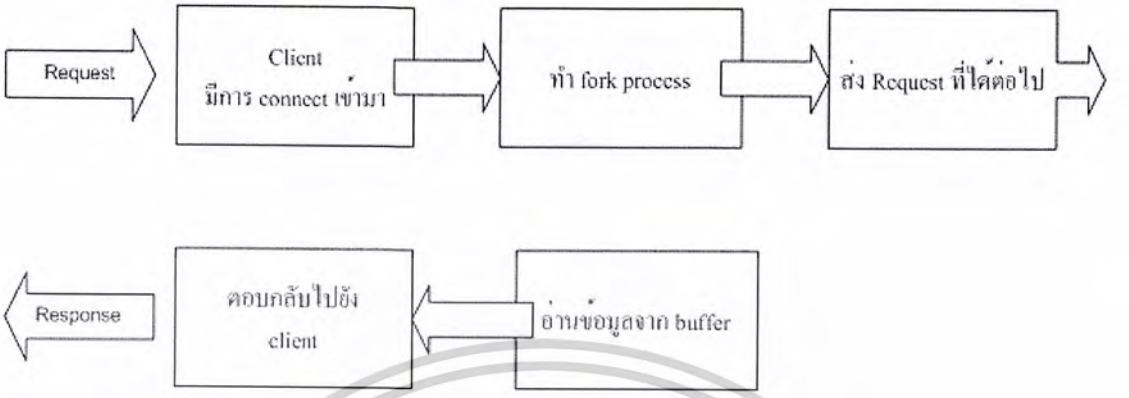


รูปที่ 3-2 แสดงอัลกอริธึมของระบบโดยรวม

จากรูปที่ 3-2 เป็นการแสดงการทำงานของระบบโดยรวม กล่าวคือ เมื่อระบบนี้เริ่มทำงานจะต้องมีการเตรียมพร้อมต่างๆ ได้แก่ การโหลดรูล (rule) เข้าไปในหน่วยความจำหลักในรูปแบบลิงค์ลิสท์ (link list) เมื่อมีการเตรียมพร้อมแล้วก็จะมีโปรเซสหลัก (main process) หรือ โปรเซสแม่คอยรับการร้องขอ (request) เมื่อมีการร้องขอเข้ามา โปรเซสแม่จะทำการฟอร์ก (fork) โปรเซสลูก (child process) ขึ้นมาเพื่อทำงานในส่วนนั้นๆ และ โปรเซสแม่จะไปคอยรับการร้องขอต่อไป เมื่อโปรเซสลูกทำงานก็จะนำการร้องขอที่เข้ามานั้นส่งไปยังส่วนตรวจสอบข้อมูลเพื่อตรวจสอบว่าการร้องขอที่เข้ามานั้นมีความปลอดภัยหรือไม่ ถ้าหากปลอดภัยก็ส่งทำการส่งต่อไปให้ยังเว็บเซิร์ฟเวอร์ เพื่อประมวนผลปกติ และเมื่อเว็บเซิร์ฟเวอร์ตอบกลับมา ก็จะส่งการตอบกลับนั้นไปที่ผู้ใช้บริการต่อไป แต่ถ้าหากส่วนกรองข้อมูลตรวจสอบได้ว่าการร้องขอที่เข้ามานั้นไม่ปลอดภัย ก็จะไม่ส่งการร้องขอที่มีลักษณะเข้าข่ายการโจมตีนั้นไปที่เว็บเซิร์ฟเวอร์เพื่อป้องกันการโจมตีนั้นๆ และจะตอบกลับไปเองในลักษณะคล้ายๆกับเว็บเซิร์ฟเวอร์ที่บอกว่าไม่มีไฟล์นั้นๆ อยู่บนเซิร์ฟเวอร์นี้ ซึ่งก็คือการตอบรหัสเอชทีทีพี 404 การกระทำดังกล่าวนี้ทำให้ผู้โจมตีไม่รู้ว่ามีระบบนี้ทำงานอยู่ตรงกลางซึ่งเป็นการทำงานแบบทรานซพาเรนท์ (transparent) และเมื่อมีการตรวจสอบได้ว่าไม่ปลอดภัยก็จะมีระบบแจ้งเตือนทั้งทางหน้าจอ ทางล็อกไฟล์ และทางฐานข้อมูล

ในส่วนของการอัพเดทนี้จะทำได้โดยการเปลี่ยนแปลงรูล ซึ่งส่วนที่ทำหน้าที่จัดการเรื่องนี้ คือ โปรแกรมแปลง (parser program) โดยหลักการทำงานของโปรแกรมแปลงนี้จะอ่านค่าจากกฎของสนอร์ท (snort rule) มาแล้วตัดเอาส่วนที่ต้องการเท่านั้น ซึ่งในที่นี้คือ ส่วนของข้อความ (message) และส่วนของรูล ที่ต้องนำไปใช้ในกระบวนการตรวจสอบต่อไป นอกจากนี้หากต้องการจะอัพเดทตัวรูลนี้ต้องใช้โปรแกรมแปลงนั้นเข้าช่วยเนื่องจากการเขียนไฟล์รูลนี้ โปรแกรมจะเขียนต่อจากไฟล์เดิม ฉะนั้นหากต้องการอัพเดทรูล นี้ก็จะต้องทำการเลือกกฎของสนอร์ทมาให้ตรงกับความต้องการมากที่สุดเพื่อใช้เป็นอินพุทของโปรแกรมแปลงนี้

3.3 ส่วนฝั่งผู้ใช้บริการ (client side)



รูปที่ 3-3 แสดงการทำงานของส่วนฝั่งผู้ใช้บริการ (client side)

ส่วนฝั่งผู้ใช้บริการ (client side) นี้เป็นส่วนที่ติดต่อกับผู้ใช้บริการ (client) ซึ่งในที่นี้ก็คือ เว็บบราวเซอร์ (web browser) นั่นเอง ส่วนฝั่งผู้ให้บริการนี้แบ่งออกเป็นสองส่วนหลักๆ ดังนี้

ส่วนที่คอยรับการเชื่อมต่อ (connection) จากรูปด้านบน เมื่อมีการร้องขอ (request) เข้ามาที่ส่วนนี้ จะทำให้มีการทำฟอร์ก (fork) โปรเซส ขึ้นคือ จะมีโปรเซสแม่ (main process) รอรับการเชื่อมต่อโดยไม่ประมวนผลแต่จะทำการฟอร์กโปรเซสลูก (child process) ขึ้นมาทำงานแทนคล้ายกับ เอชทีทีพีดี (HTTPD) ซึ่งเป็นเดมอน (demon) ของเว็บเซิร์ฟเวอร์ (web server)ทั่วไป เมื่อมี การฟอร์กโปรเซสลูกขึ้นมาทำงานต่างๆแทนในแต่ละครั้งแล้วโปรเซสแม่ ก็จะทำหน้าที่คอยรับ การเชื่อมต่อจากผู้ใช้บริการต่อไปเพื่อเป็นการทำคอนเคอร์เรนซีคอนโทรล (concurrency control) ให้รองรับผู้ใช้บริการได้มากกว่าหนึ่งในเวลาเดียวกัน ต่อจากนั้นโปรเซสลูกจะรับการร้องขอ นั้นมาแล้วส่งต่อไปยังส่วนกรองข้อมูล (filter module) เพื่อตรวจสอบความปลอดภัยของการร้องขอ นั้นต่อไป

ส่วนต่อมาจะเป็นส่วนที่ทำหน้าที่คอยรับการตอบกลับ (response) ที่ได้รับจากส่วนกรองข้อมูล ในกรณีที่การร้องขอ ที่ตรวจสอบนั้นเข้าข่ายการโจมตีหรือไม่ปลอดภัย และรับการตอบกลับ ที่ได้รับจากฝั่งผู้ให้บริการ (server side) กรณีที่การร้องขอนั้นปลอดภัย ซึ่งเป็นข้อมูลที่ได้จากเว็บเซิร์ฟเวอร์ มาเก็บไว้ที่บัฟเฟอร์ (buffer) เมื่อมีการมาแล้วจะตอบกลับไปที่ผู้ใช้บริการ หรือเว็บบราวเซอร์ (web browser) นั่นเอง

จากข้างต้นจะแสดงถึงการทำงานโดยรวม ต่อไปจะเป็นการแสดงอินพุท (input) และ เอาท์พุท (output) ของระบบ

ลักษณะของอินพุท เป็นเมสเซสก์ (message) ที่เว็บบราวเซอร์ ร้องขอเข้ามา ดังรูป

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/x-shockwave-flash, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, */*
Accept-Language: th
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: www.victim.com
Connection: Keep-Alive
```

รูปที่ 3-4 แสดงอินพุทของระบบ

จากรูปด้านบน แสดงให้เห็นถึงอินพุท (input) ที่มีลักษณะตรงตามเอชทีทีพี โปรโตคอล (HTTP protocol) ในที่นี้เป็นเฮดเดอร์ (header) ของการร้องขอ (request) การใช้งานหน้าเว็บปกติ และมีลักษณะเป็น ข้อความปกติ (plain text) ทำให้เหมาะสมต่อการใช้ อัลกอริธึม บอยเออร์มัว (boyer moore algorithm) ในการตรวจสอบความปลอดภัย ของอินพุทซึ่งก็คือ การร้องขอที่เข้ามา

ในส่วนของเอาท์พุท (output) ในที่นี้คือการตอบกลับ (response) นั้นจะเป็นลักษณะคล้ายกับอินพุท (input) แต่จะแยกได้เป็น 2 ประเภทหลัก คือเอาท์พุทปกติในกรณีที่การร้องขอ (request) ที่เข้ามานั้นปลอดภัย และ เว็บเซิร์ฟเวอร์ ตอบกลับไปจากลักษณะของ เอาท์พุทในกรณีที่เป็นการร้องขอปกติ ซึ่งในส่วนนี้เว็บบราวเซอร์ จะนำเอาท์พุท ที่ได้ไปตีความต่อไป เช่น

```
HTTP/1.1 200 OK
Date: Mon, 11 Aug 2003 11:54:50 GMT
Server: Apache/1.3.9 (Unix)
Last-Modified: Thu, 07 Aug 2003 04:42:26 GMT
ETag: "265c-319-3f31d8b2"
Accept-Ranges: bytes
Content-Length: 793
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<html>
<head>
<title>victim.com</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-874">
<bgsound src="12august/12.mid" loop="-1">
</head>
<body>
This is www.victim.com
</body>
</html>
```

รูปที่ 3-5 แสดงเอาท์พุทปกติที่ได้จากเว็บเซิร์ฟเวอร์

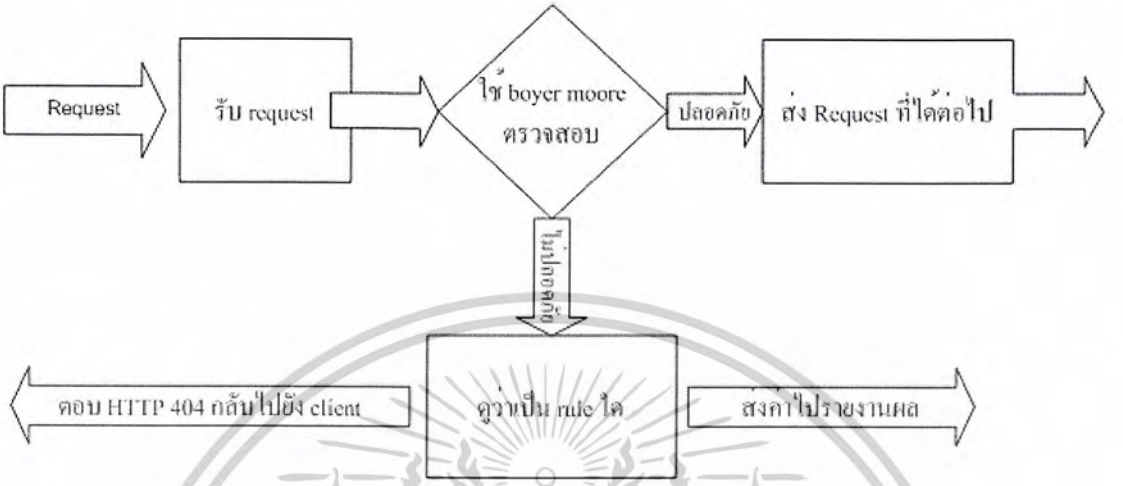
จากรูป เป็นการตอบกลับปกติ ที่มาจากเว็บเซิร์ฟเวอร์ เป็นส่วนของเอชทีทีพีเมสเซสก์ (HTTP message) ปกติ

เอาท์พุทอีกประเภทหนึ่งคือ เอาท์พุทที่ตอบมาจากส่วนกรองข้อมูล (filter module) คือ เมื่อ ส่วนกรองข้อมูลตรวจสอบได้ว่าการร้องขอไม่ปลอดภัยจะตอบกลับมาเอง โดยจะมีลักษณะเป็นรหัส 404 หรือ เอชทีทีพี 404 ซึ่งก็คือการไม่เจอไฟล์ที่ต้องการนั้น โดยเป็นการป้องกันไม่ให้ผู้โจมตีรู้ว่ามีระบบนี้ทำงานอยู่และจะไม่ส่งการร้องขอนั้นไปที่เว็บเซิร์ฟเวอร์

```
HTTP/1.1 404 Not Found
Date: Sun, 07 Sep 2003 17:33:41 GMT
Server: Apache/1.3.27 (Unix)  (Red-Hat/Linux) PHP/4.1.2 mod_throttle/3.1.2
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

11c
<HTML>
<HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD>
<BODY>
<H1>Not Found</H1>The requested URL /upload.code was not found on this server.
<P><HR>
<ADDRESS>Apache/1.3.27 Server at vv.victim.com Port 80</ADDRESS>
</BODY>
</HTML>
0
```

รูปที่ 3-6 แสดงเอาท์พุทที่ส่วนกรองข้อมูลตอบกลับไป

3.4 ส่วนกรองข้อมูล (filter module)

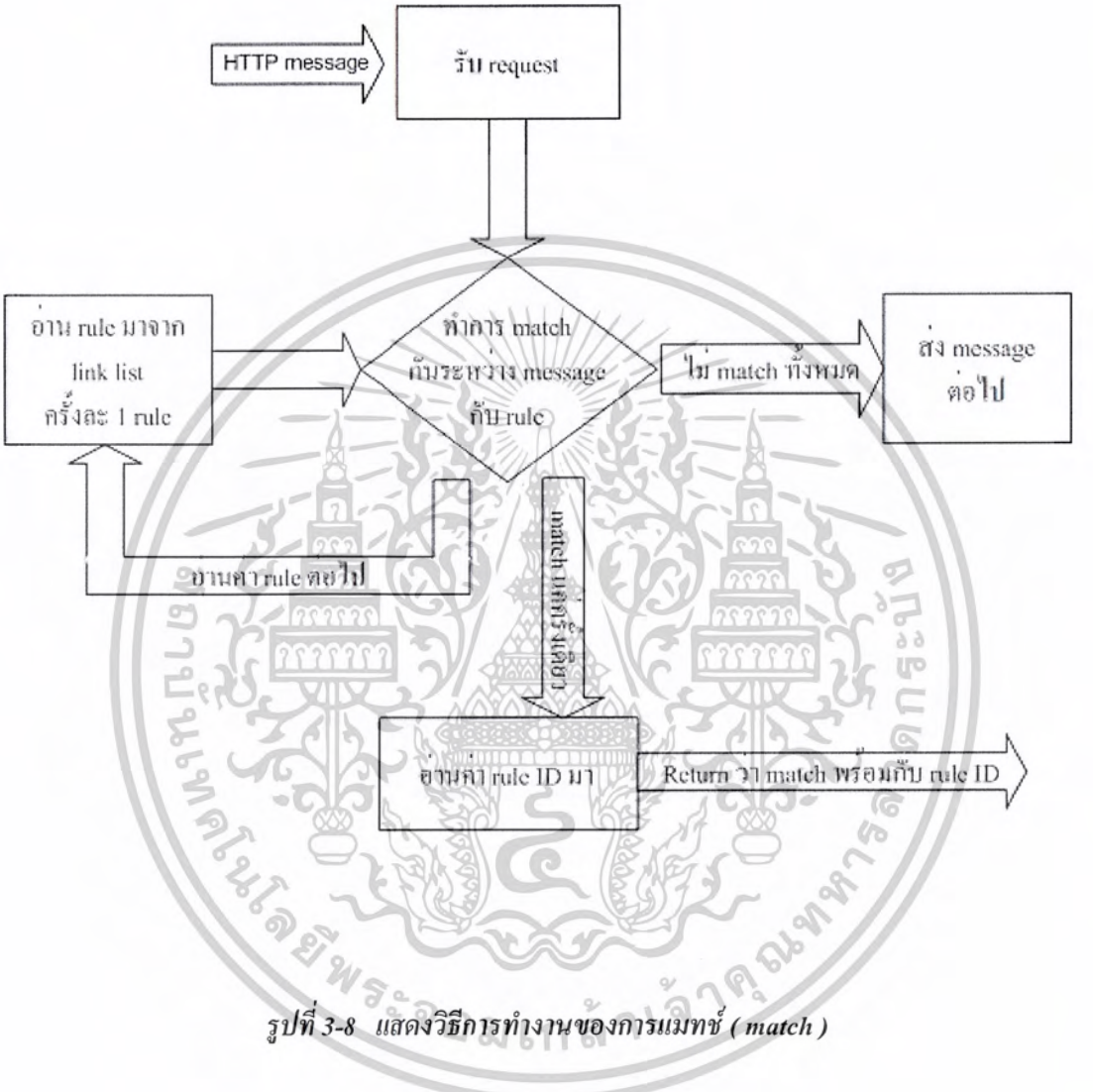


รูปที่ 3-7 แสดงการทำงานของส่วนกรองข้อมูล

ส่วนการกรองข้อมูล (filter module) นี้เปรียบเป็นส่วนสำคัญที่สุด ส่วนหนึ่งของระบบนี้ เนื่องจากระบบนี้เป็นระบบที่ตรวจสอบการร้องขอ (request) ที่เข้ามา ในส่วนการทำงานนั้น จากรูป เมื่อมีการร้องขอเข้ามา จะมีการรับไว้ จากนั้นจะนำการร้องขอนั้นเข้าวิธีการตรวจสอบโดยใช้บอยเออร์มัวร์ อัลกอริ-ธึม (boyer moore algorithm) ซึ่งเป็นอัลกอรึธึมในการทำสริงแพทเทอร์นแมทชิ่ง (string pattern matching) เริ่มจากจะมีการอ่าน รูล มาจากลิงค์ลิสท์ (linked list) ทีละหนึ่งรูล เพื่อนำมาแมทช์กับการร้องขอที่เข้ามา เมื่อตรวจสอบรูลแรกแล้วไม่แมทช์ก็จะอ่านรูล ตัวต่อไปจากลิงค์ลิสท์ และทำเช่นนี้ไปเรื่อยๆจน รูล ที่อยู่ในลิงค์ลิสท์ถูกนำมาตรวจสอบทุกตัวแล้ว หรือ อีกกรณีคือ รูลที่นำมานั้นมีการแมทช์คือไม่ปลอดภัย

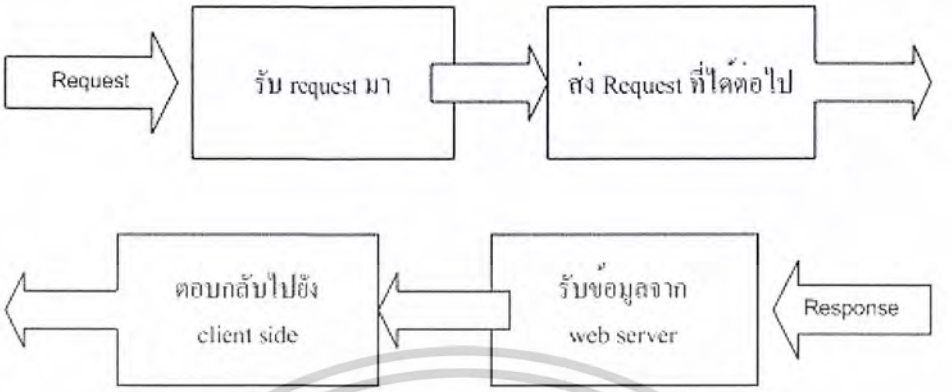
กรณีที่อ่านรูลทุกตัวแล้วไม่แมทช์ คือ ปลอดภัยนั้น ส่วนกรองข้อมูลนี้จะทำการส่งต่อ(forward) การร้องขอนั้นต่อไป

กรณีที่มีการตรวจแล้วมีการแมทช์ คือ ไม่ปลอดภัย ส่วนกรองข้อมูลนี้จะอ่านหมายเลขประจำรูล (rule ID) มาแล้วส่งหมายเลขประจำรูลนี้ไปยังส่วนแจ้งเตือนความผิดปกติต่อไป และ ยังจะมีการตอบกลับเป็น HTTP 404 ไปยังส่วนฝั่งผู้ใช้บริการ



รูปที่ 3-8 แสดงวิธีการทำงานของการแมทช์ (match)

3.5 ส่วนฝั่งผู้ให้บริการ (server side)



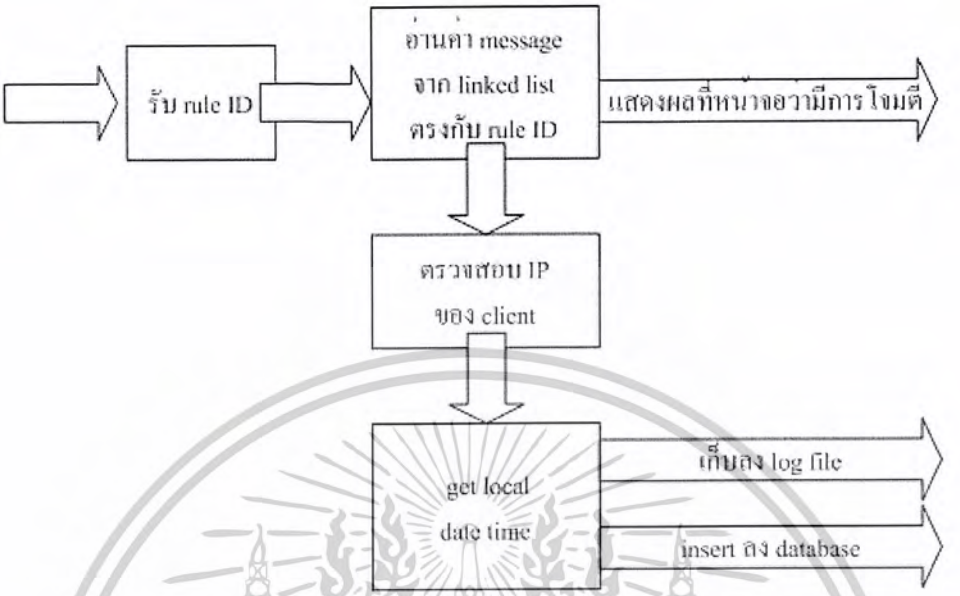
รูปที่ 3-9 แสดงส่วนฝั่งผู้ให้บริการฝั่งผู้ให้บริการ (server side)

ฝั่งผู้ให้บริการ (Server side) นี้แบ่งเป็นส่วนหลักๆ คล้ายๆกันฝั่งผู้รับบริการ(client side) คือ ส่งข้อมูล และ รับข้อมูล ซึ่งการติดต่อหลักจะเป็นการติดต่อกับเว็บเซิร์ฟเวอร์ซึ่งมีรายละเอียดดังนี้

ส่วนของการส่งข้อมูลนี้จะเป็นส่วนที่รับข้อมูลต่อมาจาก ส่วนกรองข้อมูล (filter module) ซึ่งเป็นข้อมูลการร้องขอ (request) ที่ผ่านการตรวจสอบแล้วว่าปลอดภัยสามารถส่งให้เว็บเซิร์ฟเวอร์ได้ เมื่อรับข้อมูลการร้องขอ มาโดยสมบูรณ์แล้วจะทำการส่งต่อไปให้เว็บเซิร์ฟเวอร์ประมวลผลต่อไป ซึ่งจะเห็นว่าเครื่องที่ร้องขอ ไปยังเว็บเซิร์ฟเวอร์ จริงๆ คือเครื่องที่รันระบบนี้อยู่

ส่วนของการรับข้อมูล (response) ที่ได้รับมาจากเว็บเซิร์ฟเวอร์ เมื่อเว็บเซิร์ฟเวอร์ ประมวลผลเสร็จแล้วจะส่งข้อมูลกับมายังผู้ที่ร้องขอ ซึ่งก็คือผ่านทางฝั่งผู้ให้บริการเมื่อมีการรับข้อมูลซึ่งโดยปกติแล้วจะไม่สามารถรับทั้งหมดได้ภายในครั้งเดียว เนื่องจาก เมื่อเว็บเซิร์ฟเวอร์ จะส่งก็ส่งได้เท่ากับขนาดของบัฟเฟอร์หน้าต่าง (windows size) แต่ละครั้งเท่านั้น ซึ่งเป็นกลไกของทีซีพี โปรโตคอล (tcp protocol) ตามปกติ ต้องมีการรับไปเรื่อยจนกว่าจะไม่มีข้อมูล การรับข้อมูลนี้จะรับมาแล้วส่งต่อโดยทันทีโดยไม่มีการเก็บข้อมูลไว้ (nocaching)

3.6 ส่วนแจ้งเตือนความผิดปกติ (Report module)



รูปที่ 3-10 แสดงส่วนแจ้งเตือนความผิดปกติ

ในส่วนของการรายงานผลนี้ เมื่อส่วนของการกรองข้อมูล (filter module) ตรวจสอบได้ว่าการร้องขอ (request) ที่เข้ามานั้น เข้าข่ายการโจมตี หรือ ไม่ปลอดภัย ก็จะมีการส่งหมายเลขของรูล (rule ID) ที่ตรงกับลักษณะการโจมตีนั้นๆ มาให้กับส่วนแจ้งเตือนความผิดปกติ (report module) เพื่อที่จะรายงานกับผู้ดูแลระบบต่อไป ซึ่งการรายงานจะมีอยู่ด้วยกันสามทาง คือ ส่วนการเตือนทางหน้าจอ ส่วนการเตือนทางล็อกไฟล์ (log file) และ ส่วนแจ้งลงฐานข้อมูล

ส่วนการเตือนทางหน้าจอ จะเป็นการรายงาน การโจมตีว่าเป็นการโจมตีแบบใดเข้ามาในระบบ มีรูปแบบเป็นข้อความ ดังนี้

```
Rule ID : xxx : Message from rule xxx : LOGGING!!!

Rule ID : 32 : WEB-IIS unicode directory traversal attempt : LOGGING!!!
```

รูปที่ 3-11 แสดงรูปแบบของการเตือนทางหน้าจอ

ส่วนการเตือนทางล็อกไฟล์ (log file) เป็นส่วนรายงานหลักของระบบ เนื่องจากผู้ดูแลระบบ ไม่อาจจะเช็คการโจมตีที่แจ้งเตือนผ่านทางหน้าจอได้ตลอดเวลา จึงต้องตรวจสอบล็อกไฟล์ภายหลัง โดยลักษณะของการเก็บข้อความในล็อกไฟล์นี้จะเก็บรายละเอียดอยู่สามอย่างหลักๆ ที่ผู้ดูแลระบบจะสามารถตรวจสอบที่มาของการโจมตีได้ คือ ต้องทราบว่าเป็นการโจมตีแบบใด จากเครื่องไหน และ ณ เวลาใด ซึ่งรูปแบบของการเก็บข้อมูลในล็อกไฟล์มีดังนี้

Date time / IP address / Message

รูปที่ 3-12 แสดงรูปแบบการจัดเก็บของล็อกไฟล์ (log file)

ในที่นี้จะจัดเก็บในรูปแบบของ วัน และ เวลา ที่มีการโจมตีเป็นเวลาท้องถิ่น (local time) ของเครื่อง ต่อมาเป็นหมายเลข IP address ของผู้โจมตีเข้ามา สุดท้ายเป็นส่วนของข้อความที่เตือนว่ามีการโจมตีในรูปแบบไหน

ส่วนแจ้งลงฐานข้อมูล ส่วนนี้เป็นการเพิ่มความสามารถของส่วนแจ้งเตือนความผิดปกติ ให้สามารถนำไปใช้รายงานผลในรูปแบบต่างๆได้หลากหลายขึ้น เช่น อาจจะมีการรายงานผ่านเว็บแอพพลิเคชั่น (web application) หรือ อาจจะเป็นซอร์ฟแวร์ (software) ที่ใช้ข้อมูลจากฐานข้อมูล ซึ่งความเหมาะสมน่าจะแสดงในแบบเว็บแอพพลิเคชั่น โดยจะแสดงในรูปแบบกราฟฟิค (graphic) ให้ผู้ดูแลระบบดูรายงานได้มากขึ้น ซึ่งรูปแบบที่เก็บลงฐานข้อมูลนี้จะมีรูปแบบคล้ายกับที่เก็บในล็อกไฟล์ดังนี้

ID / date time / ip / message

รูปที่ 3-13 แสดงรูปแบบการจัดเก็บของข้อมูลที่ลงฐานข้อมูล

3.7 ส่วนการปรับแต่งโปรแกรม (configuration module)

ในส่วนการปรับแต่งนี้ โดยปกติแล้วทุกระบบนี้ต้องมีการปรับแต่งอยู่แล้ว และในระบบปฏิบัติการลินุกซ์ (linux) นี้นิยมใช้การปรับแต่งในแบบ ไฟล์ (configuration file) และระบบนี้ก็ได้ใช้ไฟล์ในการปรับแต่ง เพื่อให้ระบบทำงานได้ตามความต้องการของผู้ดูแลระบบ (administrator) มากที่สุด ในส่วนของการปรับแต่งระบบนี้ จะมีรายละเอียดดังรูป

```
1 # THIS IS A CONFIG FILE OF SECURE REVERSE WEB PROXY #
2 # You must fill all of item for sure this right web server and right port #
3
4 # PORT is port of rwproxy to comunicate with client. defualt is 80 #
5 PORT:80;
6
7 # WEB_SERVER is IP address of web server that want to protect. MAX length is 30 characters #
8 WEB_SERVER:161.246.4.7;
9
10 # SERVER_PORT is port of web server that comunicate with client. defualt is 80 #
11 SERVER_PORT:80;
12
13 # FULL PATH OF FILE NAME #
14 # RULES is path of rule file that use in Boyer Moore. defualt is rule.atk #
15 RULES:rules.atk;
16
17 # LOG is path of log file that loged it when Rwproxy had intrude. defualt is rwp.log #
18 LOG:rwp.log;
19
23 # DB is flag to check "want to use LOG to MySQL ,yes or no". defualt is yes #
24 DB:no;
25
26 # Information of database #
27 # DB name to create your table #
```

```
28 ADB:rwproxy;
29
30 # DB table name that want to keep your intrude data #
31 BDB:intrude;
32
33 # DB host name that your database is running on #
34 CDB:localhost;
35
36 # DB username of host#
37 DDB:rpwroxy;
38
39 # DB passwd #
40 EDB:rpwroxy;
```

รูปที่ 3-14 แสดงไฟล์ของการปรับแต่ง

บรรทัดที่ขึ้นต้นด้วย # นี้บอกว่าเป็น comment ไม่มีผลต่อ configuration

a) ซึ่งส่วนของสิ่งที่ต้อง config ก็คือ port นั้นคือ port ของเครื่องที่ run โปรแกรม rwproxy ตัวนี้ซึ่งโดยปกติแล้วจะเป็น port 80 หรือจะเปลี่ยนเป็นหมายเลข port ที่ต้องการ

b) เว็บเซิร์ฟเวอร์เป็นส่วนที่บอกให้รู้ว่า เราต้องการให้ rwproxy นี้ป้องกันเว็บเซิร์ฟเวอร์เครื่องไหน โดยจะเปลี่ยนเป็น IP address ของเว็บเซิร์ฟเวอร์ที่ต้องการ

c) Server port นี้ เป็น port ของเว็บเซิร์ฟเวอร์ที่เราต้องการป้องกัน ซึ่งโดยปกติแล้วก็จะเป็น port 80

d) Rules นี้ บอกให้รู้ว่า rules file ที่ใช้กับ rwproxy นี้เป็น file เป็น file ไหนและอยู่ใน path ไหน

e) Log นี้ บอกให้รู้ว่า log file ที่ใช้กับ rwproxy นี้เป็น file เป็น file ไหนและอยู่ใน path ไหน

f) Db นี้ก็บอกให้รู้ว่าจะนำ log file ที่ได้ แสดงผลใน database ด้วยหรือไม่ ซึ่งเป็น MySQ

เมื่อมีการใช้ฐานข้อมูลก็จะสามารถปรับแต่งฐานข้อมูลได้

g) ADB นี้เป็นชื่อของฐานข้อมูลที่ใช้

h) BDB เป็นส่วนที่บอกให้รู้ว่าชื่อของตารางที่ใช้คืออะไร

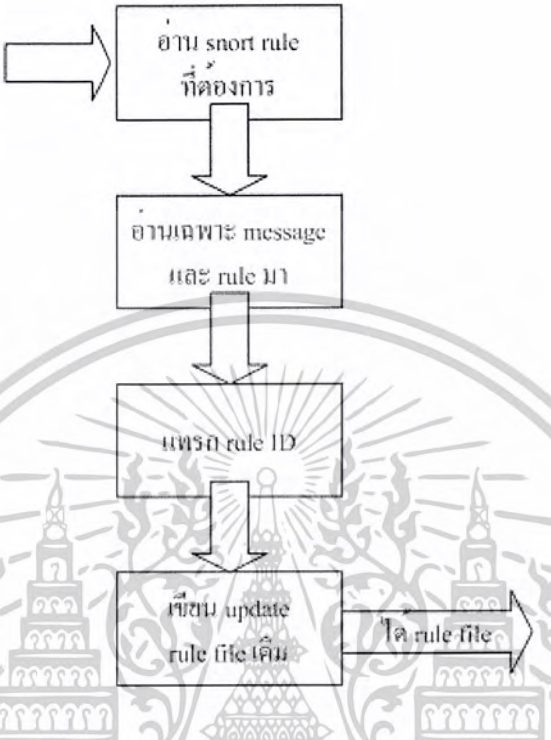
i) CDB เป็นส่วนที่บอกว่าฐานข้อมูลนี้ตั้งอยู่ที่เครื่องไหน

j) DDB เป็นส่วนที่บอกให้รู้ว่าต้องใช้ชื่อผู้ใช้ (user name) อะไรในการเข้าใช้ฐานข้อมูล

k) EDB เป็นส่วนขอรหัสผ่านของชื่อผู้ใช้นั้นๆ

3.8 ส่วนการอัพเดทและโปรแกรมแปลง (update and parser program)

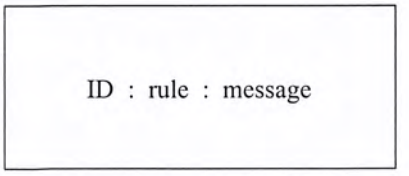
3.8.1 โปรแกรมแปลง (parser)



รูปที่ 3-15 แสดงการทำงานของโปรแกรมแปลง

โปรแกรมแปลง (Parser) นี้จะแยกออกเป็นอีกโปรแกรมหนึ่งที่ทำหน้าที่ในการนำรูล (rule) ที่นำมาจากรูลต้นฉบับของสนอร์ท (snort) เพื่อนำมาแปลงให้เป็นรูปแบบที่สามารถนำไปใช้กับระบบนี้อย่างมีประสิทธิภาพ

หลักการทำงานของโปรแกรมแปลงนี้จะทำงานดังรูป ด้านบนคือ เมื่อเลือก กฎของสนอร์ท ซึ่งจะต้องเลือกมาเฉพาะรูลที่มีความเกี่ยวข้องกับเว็บเซิร์ฟเวอร์ซึ่งวิธีการเลือกรูลนั้นจะกล่าวในเรื่องต่อไป เมื่อได้รูลที่ต้องการ โปรแกรมจะอ่านมาเฉพาะในส่วนของข้อความที่จะเตือน (message) และ ส่วนของตัวรูลเอง และจะมีการเพิ่มส่วนของหมายเลขอ้างอิงเข้าไปกับรูลด้วยเพื่อความสะดวกในการอ้างอิง เมื่อได้ครบทั้งสามอย่างดังรูปที่ 3-15 นี้ก็จะทำงานเขียนลงไฟล์รูลใหม่



รูปที่ 3-16 แสดงรายละเอียดในรูลไฟล์ (rule file)

- หลักการเลือก รูลไฟล์ (rules file) ของสนอร์ท (snort)

  เลือกเฉพาะ ที่เป็น รูล ของ เว็บ ในที่นี้จะเลือกมาใช้ทั้งหมด 6 ตัวคือ

  shellcode.rules เป็น รูล ที่ใช้ป้องกันการได้เชลล์ (shell) ที่ได้จากการทำงานที่ผิด (exploit)

  web-attacks.rules เป็น รูล ที่รวมการโจมตีต่างๆ

  web-cgi.rules เป็น รูล ที่รวมการโจมตีทางสคริปต์ซีจีไอ (cgi script)

  web-iis.rules เป็น รูล ที่รวมการโจมตีเว็บเซิร์ฟเวอร์ที่เป็นชนิดไอไอเอส (IIS) เอาไว้

  web-misc.rules เป็น รูล ที่รวมการโจมตี เว็ป ทั่วไป

  web-php.rules เป็น รูล ที่รวม รูปแบบของ พีเอชพี (php) ที่สามารถใช้ โจมตีได้ ซึ่ง รูล นี้มีปัญหาค่อนข้างมาก ต้องอาศัยความชำนาญในการ เลือกเฉพาะ รูล ที่จำเป็นจริงๆ เท่านั้นมาใช้ เพราะ ถ้านำมาทั้งหมดจะไม่สามารถใช้ file.php ได้เลย

ที่เลือกมานี้เป็น รูล เฉพาะการโจมตี เว็บเซิร์ฟเวอร์ เท่านั้น ในส่วนของ รูลไฟล์ อื่นๆ ที่ไม่เกี่ยวกับเว็บ นั้นไม่ควรนำมาใช้ เนื่องจากระบบนี้จะนำไปใช้ในการตรวจสอบการร้องขอ (request) เว็บเท่านั้น ถ้าเพิ่ม รูล อื่นๆ เข้ามาก็จะไม่สามารถนำไปใช้ประโยชน์ได้ เป็นการทำให้สิ้นเปลืองโดยใช่เหตุ

3.8.2 การอัพเดท

ในส่วนของการอัพเดท (update) การอัพเดทตัวระบบนี้จะทำได้โดยการอัพเดทส่วนของรูลใหม่เท่านั้นคือ ต้องมีการใช้โปรแกรมแปลงเพิ่มส่วนของรูลเข้าไปในไฟล์ของรูล หรือไม่ต้องสร้างไฟล์รูลใหม่ทั้งหมดเพื่อความมีประสิทธิภาพ

- การ อัพเดท ไฟล์รูล ภายหลัง เช่นกรณีที่ สนอร์ท (snort) ออก รูล ใหม่

  ข้อแนะนำในการอัพเดท ควรจะลบ ไฟล์รูล ตัวเก่าทิ้งไปก่อนแล้วทำใหม่ทั้งหมดจะดีที่สุด เนื่องจาก ถ้านำ รูล ใหม่มาเพิ่มเติมกับ รูล ของระบบตัวเก่า จะทำให้มีการซ้ำซ้อนของ รูล ขึ้นได้ นอกจากนี้เมื่อมีการนำ รูล ไปใช้ทำงาน รูล ที่ใหม่กว่าจะอยู่ด้านล่าง หาก รูล ที่ใหม่กว่ามีการปรับปรุงจาก รูล เก่า แล้วมีทั้งสอง รูล อยู่ใน รูล เดียวกัน รูล ที่เก่ากว่าจะถูกนำไปใช้ก่อนเนื่องจากอยู่ด้านบน ทำให้อาจมีปัญหาได้

ดังนั้น เมื่อมีการอัพเดทรูล ก็ควรจะทำ ไฟล์รูลใหม่ทั้งหมดโดยใช้โปรแกรมแปลงเป็นตัวสร้างไฟล์รูลตัวใหม่

บทที่ 4

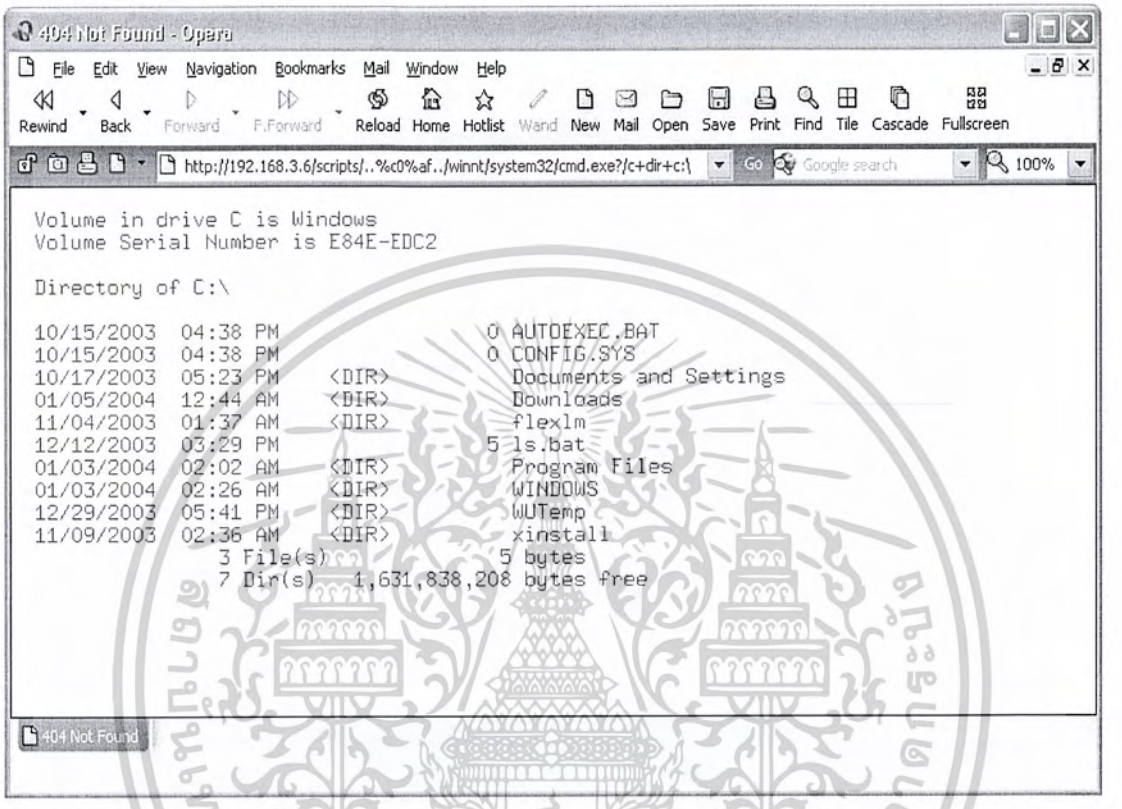
ผลการทดลอง

4.1 ผลการทดลองส่วนฝั่งผู้ใช้บริการ

เมื่อมีการร้องขอไปที่ระบบนี้ ถ้ามีการรันระบบนี้อยู่จะแสดงหน้าเว็บที่ได้ทำการคอนฟิกกูเรชั่นไว้ ในที่นี้คือเว็บของภาควิชา

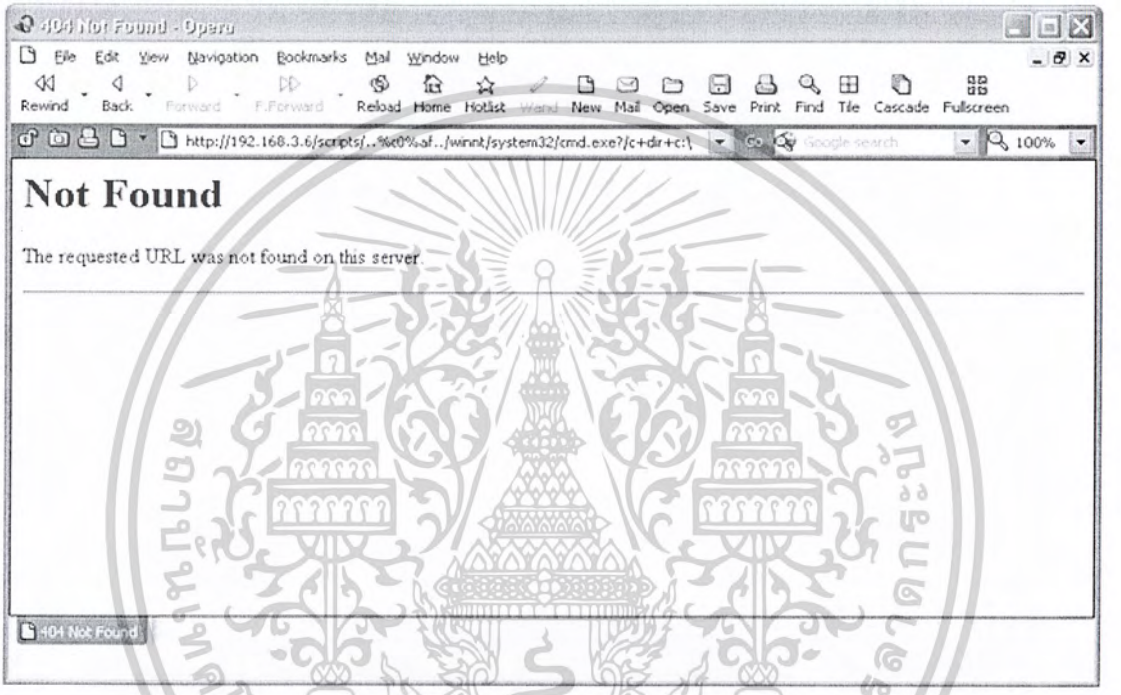
รูปที่ 4-1 แสดงหน้าเว็บภาควิชาที่เรียกผ่านรีเวิร์สเว็บพรอกซี่แบบปลอดภัย (secure reverse web proxy)

ในกรณีที่ ยังไม่มีการนำระบบนี้มาใช้ ผู้บุกรุกสามารถเข้าถึงเว็บเซิร์ฟเวอร์ได้โดยตรง ดังรูปที่ 4-2 นี้ ผู้บุกรุกใช้วิธีการเข้าดูข้อมูลในไดเรกทอรี่ (directory traversal) ของ IIS ซึ่งจะแสดงให้เห็นข้อมูลบนไดรว์ ซี (drive C:) ทั้งหมด



รูปที่ 4-2 เมื่อไม่มีการรันโปรแกรมนี้อยู่

ในส่วนต่อมาเมื่อมีการบุกรุกเข้ามาเหมือนกรณีด้านบนนั้น ผู้บุกรุกจะต้องผ่านมาทาง ระบบนี้ก่อน ซึ่งในตัวโปรแกรมมีส่วนที่ทำหน้าที่ตรวจสอบการบุกรุกตรงนี้อยู่แล้ว ดังนั้นผู้บุกรุกจะไม่สามารถผ่านเข้าไปยัง เว็บเซิร์ฟเวอร์จริงๆ ที่อยู่ หลัง ระบบนี้ และ เมื่อตรวจสอบได้ว่าการร้องขอนั้นเป็นการบุกรุก ระบบนี้จะเป็นฝ่ายตอบกลับไปว่า มันไม่มีไฟล์ที่ร้องขอเข้ามา (404 Not Found) แทน



รูปที่ 4-3 แสดงการตอบกลับเมื่อมีการโจมตีขณะที่มีการรันโปรแกรมนี้อยู่

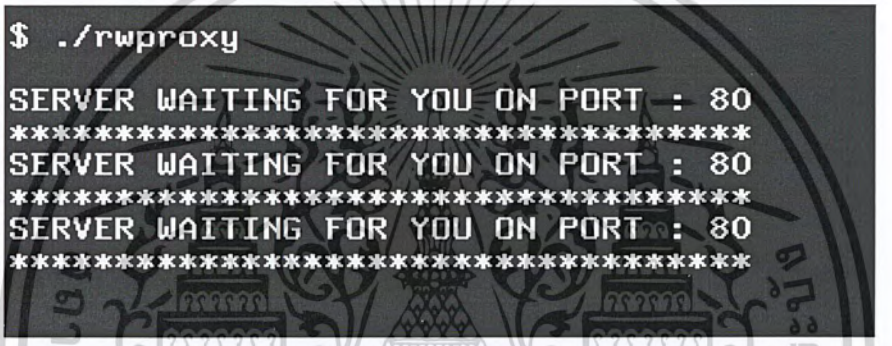
4.2 ผลการทดลองส่วนแจ้งเตือนความผิดปกติ

ผลการทดลองได้แบ่งออกเป็นสามส่วนหลักๆ คือ ส่วนของ หน้าจอ , ล็อกไฟล์ , ฐานข้อมูล

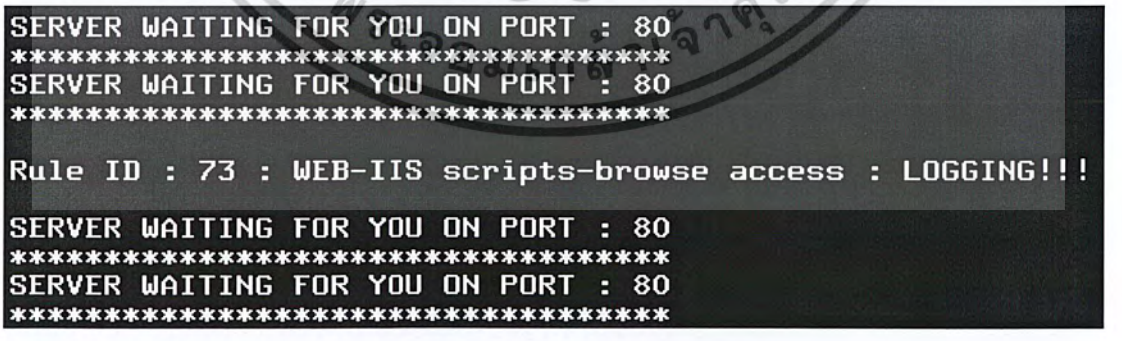
4.2.1 ส่วนของหน้าจอ

เมื่อมีการสั่งรันโปรแกรมรีเวิร์สเว็บพร็อกซี่แบบปลอดภัย (secure reverse web proxy) นี้จะมีผลต่อการแสดงผลที่หน้าจอทั้งการทำงานปกติและไม่ปกติ

เมื่อรันคำสั่ง $ ./rwproxy



รูปที่ 4-4 แสดงเอาท์พุทที่หน้าจอ เมื่อทำงานปกติ



รูปที่ 4-5 แสดงเอาท์พุทที่หน้าจอ เมื่อมีการโจมตี

4.2.2 ส่วนของล็อกไฟล์

เมื่อมีการโจมตีเข้ามาให้ระบบจะมีการเก็บลงล็อกไฟล์ ซึ่งรูปแบบของล็อกไฟล์ดังนี้

Date time | source IP | message

ล็อกไฟล์จริงมีรูปแบบ ดังรูป

```
31/01/2004 18:58:53     161.246.5.41     50 : WEB-IIS cmd.exe access
31/01/2004 18:59:39     161.246.5.41     50 : WEB-IIS cmd.exe access
31/01/2004 18:59:44     161.246.5.41     73 : WEB-IIS scripts-browse access
31/01/2004 18:59:44     161.246.5.41    395 : WEB-CGI /cgi-bin/ access
31/01/2004 18:59:44     161.246.5.41     89 : WEB-IIS iissamples access
31/01/2004 18:59:44     161.246.5.41    542 : WEB-MISC Cisco IOS HTTP configura
tion attempt
31/01/2004 18:59:44     161.246.5.41    395 : WEB-CGI /cgi-bin/ access
31/01/2004 18:59:44     161.246.5.41     89 : WEB-IIS iissamples access
31/01/2004 18:59:44     161.246.5.41    542 : WEB-MISC Cisco IOS HTTP configura
tion attempt
31/01/2004 18:59:44     161.246.5.41      4 : WEB-IIS .asp Transfer-Encoding\:
chunked
31/01/2004 18:59:44     161.246.5.41      4 : WEB-IIS .asp Transfer-Encoding\:
chunked
31/01/2004 18:59:45     161.246.5.41    702 : WEB-MISC http directory traversal
31/01/2004 18:59:45     161.246.5.41    394 : WEB-CGI db2www access
31/01/2004 18:59:45     161.246.5.41    702 : WEB-MISC http directory traversal
31/01/2004 18:59:45     161.246.5.41    249 : WEB-CGI phf arbitrary command exe
cution attempt
31/01/2004 18:59:45     161.246.5.41     50 : WEB-IIS cmd.exe access
31/01/2004 18:59:45     161.246.5.41     50 : WEB-IIS cmd.exe access
31/01/2004 18:59:45     161.246.5.41    702 : WEB-MISC http directory traversal
31/01/2004 18:59:45     161.246.5.41    702 : WEB-MISC http directory traversal
31/01/2004 18:59:45     161.246.5.41    702 : WEB-MISC http directory traversal
31/01/2004 18:59:45     161.246.5.41    702 : WEB-MISC http directory traversal
"rup.log" 1265L, 92519C                                       1,1           Top
```

รูปที่ 4-6 แสดงล็อกไฟล์ของระบบ

4.3 ผลการทดลองส่วนโปรแกรมแปลง (parser)

ผลการทดลองในส่วนนี้คือเมื่อทำการทดลองรันโปรแกรมแปลง

ในการสั่งให้โปรแกรมทำงานนั้นต้องสั่งให้ตามซินแทกซ์ (syntax) คือต้อง

```
Parser  < input file >  < output file >
```

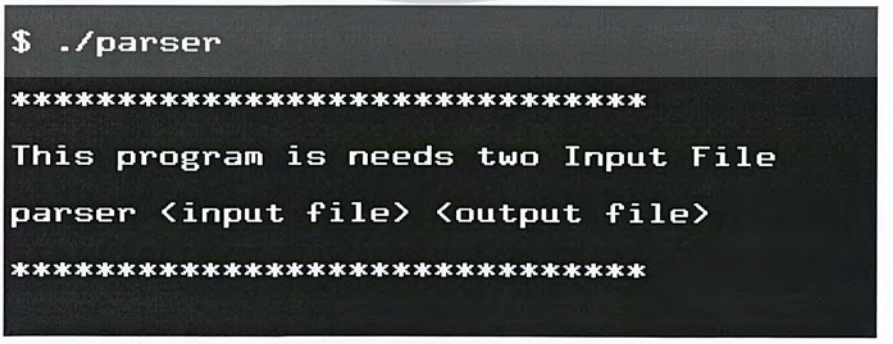
คำสั่งต่อไปนี้เป็นคำสั่งที่ใช้แปลงรูล (rule) ของสนอร์ท (snort) ที่ชื่อ web-iis.rules มาให้เป็นรูลของระบบนี้เป็นไฟล์ที่ชื่อ rule.atk

เมื่อรันคำสั่ง $ ./parser web-iis.rules rule.atk

รูปที่ 4-7 แสดงการทำงานที่ถูกต้องของโปรแกรมแปลง

หากทำกาสั่งรันโดยที่ไฟล์อินพุท (input file) นั้นไม่มีอยู่จริง โปรแกรมจะไม่มีการทำงานต่อจะออกจากโปรแกรมทันที ดังนี้

เมื่อรันคำสั่ง $ ./parser



รูปที่ 4-8 แสดงการทำงานโดยมีอาร์กิวเมนท์ (argument) ไม่เพียงพอ

เมื่อรันคำสั่ง $ ./parser file.rules rule.atk

```
$ ./parser file.rules rule.atk
********************************
Input File Error
Please check input file.
********************************
```

รูปที่ 4-9 แสดงการทำงานของโปรแกรมแปลงเมื่อไฟล์ของรูลนั้นไม่มีอยู่จริง

เมื่อใช้โปรแกรมแปลงแล้ว ผลที่ได้คือไฟล์ของรูลที่แปลงแล้ว สามารถนำไปใช้กับระบบนี้ได้ ซึ่งรูปแบบของรูลจะมีดังนี้

Rule ID : “ rule “ : “ message “

ตัวอย่างเช่น

```
16:"LOCK ":"WEB-IIS WebDAV file lock attempt"
17:".printer":"WEB-IIS ISAPI .printer access"
18:".ida?":"WEB-IIS ISAPI .ida attempt"
19:".ida":"WEB-IIS ISAPI .ida access"
20:".idq?":"WEB-IIS ISAPI .idq attempt"
21:".idq":"WEB-IIS ISAPI .idq access"
22:"%2e.asp":"WEB-IIS %2E-asp access"
23:"/*.idc":"WEB-IIS *.idc attempt"
24:"|2e2e5c2e2e|":"WEB-IIS ..\.. access"
25:".asp|3a3a|$DATA":"WEB-IIS .asp\:\:$DATA access"
26:".bat?":"WEB-IIS .bat? access"
```

รูปที่ 4-10 แสดงรูปแบบจริงของรูลที่ใช้ในระบบนี้

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการดำเนินงาน

โปรแกรมรีเวิร์สเว็บพร็อกซี่แบบปลอดภัยนี้สามารถทำงานได้ตามวัตถุประสงค์เป็นอย่างดี คือ เน้นความปลอดภัยของเว็บเซิร์ฟเวอร์เป็นหลัก ซึ่ง feature อื่น ๆ ของรีเวิร์สเว็บพร็อกซี่นั้นไม่สามารถที่จะพัฒนาให้สามารถใช้งานได้ เนื่องจากมีความซับซ้อนเป็นอย่างมากเช่น caching ข้อมูลที่เป็นแบบ static และส่งให้กับเครื่องไคลเอ็นแทนเครื่องเว็บเซิร์ฟเวอร์ซึ่งจะเป็นการแบ่งเบาภาระลงได้เป็นอย่างมาก หรือแม้กระทั่งการทำ load balancing ที่สามารถกระจาย load ให้แก่เซิร์ฟเวอร์หลาย ๆ ตัวซึ่งตัว รีเวิร์สเว็บพร็อกนี้จะต้องค่อยดูแล session ของไคลเอ็นต์ให้สามารถทำงานได้แม้ว่าจะมีเครื่องเซิร์ฟเวอร์หลายเครื่องก็ตามดังนั้นผู้พัฒนาจึงพัฒนาให้ตัวโปรแกรมสามารถที่จะบันทึก log ลงในฐานข้อมูลได้โดยประโยชน์จากการทำเช่นนี้จะช่วยให้ผู้ดูแลระบบนั้นสามารถที่จะตรวจสอบ log ได้จากเครื่องที่เป็นฐานข้อมูลเพียงเครื่องเดียว แม้ว่าจะมีโปรแกรมรีเวิร์สเว็บพร็อกซี่แบบปลอดภัยทำงานในหลาย ๆ จุดก็ตาม ทั้งนี้ก็ยังสามารถเก็บทำสถิติได้สะดวกมากขึ้นอีกด้วย

5.2 แนวทางการพัฒนาต่อและข้อเสนอแนะ

โปรแกรมรีเวิร์สเว็บพร็อกซี่แบบปลอดภัยนี้จะเน้นในส่วนความปลอดภัยเป็นหลักเท่านั้น ซึ่งหากจะให้มีความสมบูรณ์มากขึ้นนั้นควรจะพัฒนาในส่วนที่ทำหน้าที่โหลดบาลานซ์ (Load balance) และส่วนการแคช (Cache) ข้อมูลได้ด้วย ซึ่งทั้งสองส่วนนี้เป็นส่วนจำเป็นสำหรับรีเวิร์สพร็อกซี่

ในการทำงานจริงนั้นโปรแกรมเว็บพร็อกซี่แบบปลอดภัยจะต้องทำงานร่วมกับ router , firewall และอุปกรณ์อื่น ๆ เพื่อทำให้การทำงานของโปรแกรมนั้นเป็นไปในลักษณะของ transparent โดยเมื่อมี request เข้ามาเป็น HTTP โปรโตคอล จึงค่อยนำไปให้โปรแกรมเว็บ พร็อกซี่แบบปลอดภัยตรวจสอบว่าเป็นการโจมตีหรือไม่ แต่ถ้าเป็นโปรโตคอลอื่น ก็ให้ส่งไปยังเครื่องของ server โดยตรงหรือว่าพร็อกซี่ตัวอื่น ๆ อีกก็ได้ สำหรับข้อมูลในขาออกนั้นอาจจะต้องมีการเปลี่ยนแปลง header ในส่วนของผู้ส่งแทนที่จะเป็นตัวพร็อกซี่เอง ให้เป็น IP ของเว็บเซิร์ฟเวอร์นั้นแทน

บรรณานุกรม

หนังสืออ้างอิง

[1] Jeol Scambray, Strart McClure, George Kurtz: "Hacking Exposed : Network Security Secret & Solutions", McGraw-Hill, Second Edition, 2001

[2] W. Richard Stevens: "UNIX Network Programming Volume 1", Prentice-Hall PTR, Second Edition, 1998

[3] Joel Scambray, Mike Shema: "Hacking Exposed Web Applications", McGraw-Hill,2002

[4] Neil Matthew, Richard Stones: "Beginning Linux Programming", Wrox Press, Second Edition,1999

เว็บไซต์อ้างอิง

[1] Snort NIDS http://www.snort.org

[2] Security Focus http://www.securityfocus.com

[3] ThaiCert http://thaicert.nectec.or.th

[4] http://www.sans.org

[5] Securiteam http://www.securiteam.com

[6] http://www.kernel.org

[7] http://www.debian.org