การโจมตีเว็บเซิร์ฟเวอร์ด้วยเอสคิวแอลอินเจ็คชั่นผ่านเอชทีทีพีเฮดเดอร์และแนวทางป้องกัน

SQL Injection Attack via HTTP Header and Defense Mechanism

ปิยะ อาจหาญ[1]
ดร.ชัยพร เขมะภาตะพันธ์[2]

บทคัดย่อ

สารนิพนธ์นี้นำเสนอรูปแบบของการเจาะระบบเว็บแอพพลิเคชั่นด้วยเทคนิค SQL Injection ตลอดจนวิธีการหลบหลีกการตรวจจับและป้องกัน โดยเฉพาะอย่างยิ่ง SQL Injection ผ่าน HTTP Header ซึ่งเป็นเทคนิคนึงที่ผู้บุกรุกสามารถใช้เป็นวิธีในการหลบหลีกการตรวจจับและป้องกันของระบบที่ใช้ป้องกัน โดยสารนิพนธ์นี้เสนอแนวทางการป้องกันโดยใช้ Web Application Firewall (WAF) ทำงานรวมกับ Stateful Firewall เพื่อแก้ไขปัญหาดังกล่าว

จากผลการศึกษาซึ่งแบ่งออกเป็น 3 กรณีพบว่า กรณีที่หนึ่งหากมีเพียง Stateful Firewall จะไม่สามารถป้องกันได้เลย ต่อมาเป็นกรณีที่สองมี Web Application Firewall (WAF) ป้องกัน ก็จะสามารถป้องกันระบบได้ในระดับปานกลาง ทั้งนี้นั้นขึ้นอยู่กับ Signature ที่มีและค่า Configuration ของระบบนั้น ๆ กรณีที่สามนั้นเป็นการให้ Stateful Firewall และ Web Application Firewall (WAF) ทำงานร่วมกันโดยเมื่อมีการพยายามบุกรุกอย่างต่อเนื่องและ Web Application Firewall (WAF) สามารถตรวจพบก็จะสั่งงานให้ Stateful Firewall ทำงานในการป้องกันความพยายามในการบุกรุกแทน ซึ่งทำให้การป้องกันระบบนั้นมีประสิทธิภาพสูงกว่าในกรณีที่หนึ่งและกรณีที่สอง

ABSTRACT

This Master Project demonstrates a web application penetration using SQL Injection technique and suggests the mechanism to avoid and pass through detection. Focusing on the SQL Injection through HTTP Header, one of the most effective methods to get through the security defense, this Master Project proposes the way to prevent this form of attack using a cooperation of Web Application Firewall and Stateful Firewall.

The result of this study shows that only Stateful Firewall does not have enough protection capability against this form of attack. Similarly, sole Web Application Firewall can only defend the particular attack depending on the configuration setting and the signature database.

---

[1] นักศึกษาปริญญาโท สาขาวิชาวิศวกรรมคอมพิวเตอร์และโทรคมนาคม มหาวิทยาลัยธุรกิจบัณฑิตย์

[2] อาจารย์ที่ปรึกษาวิทยานิพนธ์

However, the combination of these two types of firewall work together and form a concrete defense mechanism against this kind of penetration technique.

บทนำ

1.1ที่มาและความสำคัญของปัญหา

ในปัจจุบันมีการพัฒนาระบบที่ให้บริการจำนวนมากผ่านเว็บแอพพลิเคชั่น ซึ่งนักพัฒนาระบบยังขาดความรู้ ความเข้าใจในการพัฒนาให้ระบบให้มีความปลอดภัย ตามมาตรฐานการพัฒนาที่หน่วยงานบริษัท องค์กรต่างๆยอมรับ เช่น มาตรฐาน OWASP (Open Web Application Security Project เวอร์ชั่น ค.ศ. 2013

ซึ่งทาง OWASP ได้จัดอันดับ TOP 10 โดยภัยคุกคามที่เกิดผลกระทบสูงสุดในส่วนของเว็บแอพพลิเคชั่น พบว่าภัยคุกคามในส่วนของ A1-Injection นั้นเป็นภัยคุกคามระดับสูงสุดทุกรอบการเปลี่ยนแปลงที่ทาง OWASP ทำการเก็บสถิติตามรายละเอียดดังนี้

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

ภาพที่ 1.1 ผลกระทบที่เกิดกับเว็บแอพพลิเคชั่น เปรียบเทียบ ค.ศ. 2010 และ ค.ศ. 2013

จากภาพที่ 1.1 จะสังเกตุได้ว่าการพัฒนาเว็บแอพพลิเคชั่นจะมีการถูกโจมตีจากผู้ไม่ประสงค์ดีในส่วนการ Injection เป็นจำนวนมาก เพราะผู้พัฒนายังขาดความรู้ความเข้าใจในการป้องกันภัยคุกคามในลักษณะนี้ ในงานวิจัยนี้จึงนำเรื่องของภัยคุกคามในส่วนของการทำ Injection มาเป็นปัญหาสำคัญเพื่อที่จะให้ผู้พัฒนาเกิดความเข้าใจ และนำเครื่องมือที่มีมาใช้ตรวจสอบช่องโหว่ รวมถึงการป้องกันเพื่อลดภัยคุกคามที่จะเกิดขึ้นต่อระบบให้บริการได้

1.2ปัญหานำวิจัย

ระบบเว็บแอพพลิเคชั่นที่ใช้งานส่วนใหญ่มีช่องโหว่ในส่วนของ SQL Injection และต้องดำเนินการตรวจสอบ ลดผลกระทบที่จะเกิดขึ้นกับระบบเว็บแอพพลิเคชั่น เพื่อให้ระบบที่ให้บริการมีความปลอดภัยตามมาตรฐาน OWASP 2013 ในส่วนของ A1 Injection

1.3วัตถุประสงค์

ขอบเขตของงานวิจัยนี้ มุ่งเน้นการศึกษาการเจาะระบบเว็บแอพพลิเคชั่นโดยอาศัยช่องโหว่แบบ SQL Injection เพื่อมองหาข้อผิดพลาดต่าง ๆ ที่น่าจะเป็นจุดอ่อนของระบบรักษาความปลอดภัยที่แตกต่างกันไปตามสภาพแวดล้อมที่ไม่เหมือนกัน เพื่อให้เห็นว่าระบบรักษาความปลอดภัยที่จำเป็นในปัจจุบันนั้น ควรมีการออกแบบระบบที่ต้องประกอบไปด้วยอุปกรณ์รักษาความปลอดภัยใดบ้าง ในการตรวจจับและป้องกันการเจาะระบบแอพพลิเคชั่นด้วยช่องโหว่แบบ SQL Injection เพื่อเป็นแนวทางในการออกแบบด้านการรักษาความปลอดภัยระบบคอมพิวเตอร์เครือข่าย

1.4ขอบเขตของการวิจัย

1.4.1 ศึกษาข้อมูลเกี่ยวกับภัยคุกคามในส่วนของ SQL Injection ตามมาตรฐาน OWASP

1.4.2 ติดตั้งระบบจำลองในส่วนของระบบเว็บแอพพลิเคชั่น

1.4.3 ตรวจสอบหาช่องโหว่

1.4.4 ศึกษาข้อมูลเกี่ยวกับการลดภัยคุกคาม SQL Injection ตามมาตรฐาน OWASP

1.4.5 ออกแบบระบบเว็บเซิร์ฟเวอร์เพื่อป้องกันการโจมตีเว็บแอพพลิเคชั่น ในส่วนภัยคุกคาม SQL Injection ตามมาตรฐาน OWASP

1.4.6 สรุปผลการดำเนินงาน และทำข้อแนะนำในการพัฒนาระบบเว็บเซิร์ฟเวอร์เพื่อป้องกันในส่วนภัยคุกคาม SQL Injection

1.5ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 มีความรู้และเข้าใจถึงรูปแบบภัยคุกคามในส่วนของ SQL Injection

1.5.2 สามารถใช้เครื่องมือในการตรวจสอบหาช่องโหว่ SQL Injection

1.5.3 สามารถทราบถึงวิธีการใช้งานของเครื่องมือที่นำมาใช้ในการตรวจสอบหาช่องโหว่

1.5.4 สามารถทราบถึงระบบการป้องกันการโจมตีเว็บแอพพลิเคชั่น

1.5.5 สามารถทราบถึงแนวทางการออกแบบและเลือกติดตั้งใช้งานระบบรักษาความปลอดภัยที่ให้ระบบเว็บแอพพลิเคชั่น นั้นลดความเสี่ยงของการถูกโจมตีในลักษณะ SQL

ระเบียบวิธีวิจัย

2.1 แนวทางการวิจัยและพัฒนา

การเจาะระบบเว็บไซต์ด้วยเทคนิค SQL Injection นั้นเป็นช่องโหว่ที่เกิดขึ้นมานานแล้ว อีกทั้งความรุนแรงก็ยังอยู่ในระดับสูง แต่เว็บแอพพลิเคชั่นที่ถูกพัฒนาให้ใช้งานได้ในปัจจุบันนั้น กลับยังเป็นจำนวนไม่น้อยที่ยังพบว่ามีช่องโหว่ประเภทนี้อยู่

วิธีการที่ถูกต้องและดีที่สุดในการแก้ไขปัญหาช่องโหว่ประเภทนี้ คือ การออกแบบระบบเว็บแอพพลิเคชั่นที่จำเป็นจะต้องเชื่อมต่อกับระบบฐานข้อมูล และต้องพัฒนาซอร์สโค้ดของเว็บแอพพลิเคชั่นนั้น ๆ ให้มีความปลอดภัยมากที่สุดเท่าที่จะทำได้ แต่การทำงานทั้งกระบวนการที่กล่าวมาข้างต้นนั้น อาจไม่สอดคล้องกับปัจจัยในหลาย ๆ ด้านอาทิเช่น ทรัพยากรระบบคอมพิวเตอร์ทางด้านฮาร์ดแวร์หรือ

ซอฟต์แวร์ไม่เพียงพอ, ความรู้ความชำนาญด้านการรักษาความปลอดภัยของผู้พัฒนาเว็บแอพพลิเคชั่นตลอดจนผู้ดูแลระบบไม่ดีพอ, ผู้บริหารตลอดจนเจ้าหน้าที่ที่รับผิดชอบขาดความตระหนักด้านความมั่นคงปลอดภัยระบบคอมพิวเตอร์ ฯลฯ ซึ่งที่กล่าวมาข้างต้นนั้นเป็นสาเหตุที่ทำให้เกิดช่องโหว่ของระบบได้

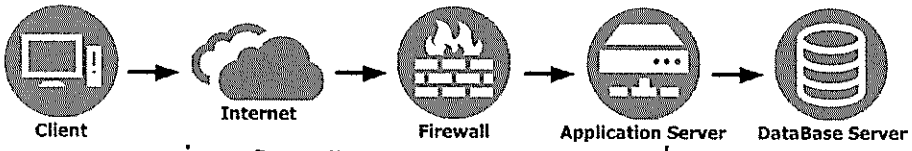
ในส่วนการป้องกันภัยคุกคามที่เกี่ยวข้องกับระบบเว็บแอพพลิเคชั่นนั้น Web Application Firewall (หรือที่เรียกกันย่อ ๆ ว่า "WAF") ถูกออกแบบมาเพื่อปกป้องการคุกคามในด้านนี้โดยเฉพาะ ซึ่งปัจจุบันก็มีทั้งที่เป็นฮาร์ดแวร์และซอฟต์แวร์ ที่ผู้ใช้งานนั้นสามารถหามาใช้งานได้ ซึ่งก็มีทั้งแบบที่ต้องมีค่าใช้จ่ายและก็มีแบบที่ไม่ต้องเสียค่าใช้จ่ายด้วยเช่นกัน

ในงานวิจัยนี้จะได้ทำการทดสอบเจาะระบบเว็บไซต์ด้วยเทคนิค SQL Injection แบบ Union Based เพื่อต้องการให้เห็นถึงระบบเว็บแอพพลิเคชั่นที่มีช่องโหว่ประเภทนี้อยู่จริง และทดสอบว่ารูปแบบของการเจาะระบบเว็บไซต์ด้วยเทคนิค SQL Injection นั้นมีโอกาสที่จะหลุดลอดจากการตรวจจับของระบบรักษาความปลอดภัยในบางระบบได้

2.2 วิเคราะห์ระบบรักษาความปลอดภัยและทดสอบเจาะระบบเว็บไซต์ด้วยเทคนิค SQL Injection

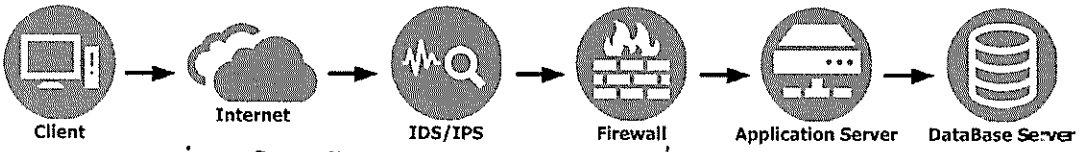
ผู้วิจัยได้ทำการทดสอบโดยใช้ช่องทางการเชื่อมต่อระบบเครือข่ายคอมพิวเตอร์ผ่านระบบอินเตอร์เน็ตเท่านั้น อันเนื่องด้วยวิธีวิจัยที่ต้องการจะทดสอบให้เห็นว่าระบบคอมพิวเตอร์ทั่ว ๆ ไปนั้นส่วนใหญ่ก็มักจะมีอุปกรณ์รักษาความปลอดภัยอย่าง IDS/IPS, Packet Filtering Firewall, Stateful Inspection Firewall, Application Proxy หรือ Hybrid Firewall ติดตั้งไว้ในระบบเครือข่ายคอมพิวเตอร์และถูกวางขวางเครื่องคอมพิวเตอร์เว็บเซิร์ฟเวอร์นั้นอยู่แล้ว

รูปแบบการติดตั้งโครงสร้างระบบรักษาความปลอดภัยที่มี Firewall



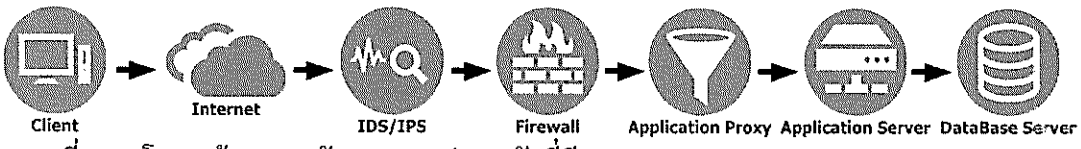
ภาพที่ 2.1 โครงสร้างระบบรักษาความปลอดภัยที่มี Firewall

รูปแบบการติดตั้งโครงสร้างระบบรักษาความปลอดภัยที่มี Firewall และ IDS/IPS



ภาพที่ 2.2 โครงสร้างระบบรักษาความปลอดภัยที่มี Firewall และ IDS/IPS

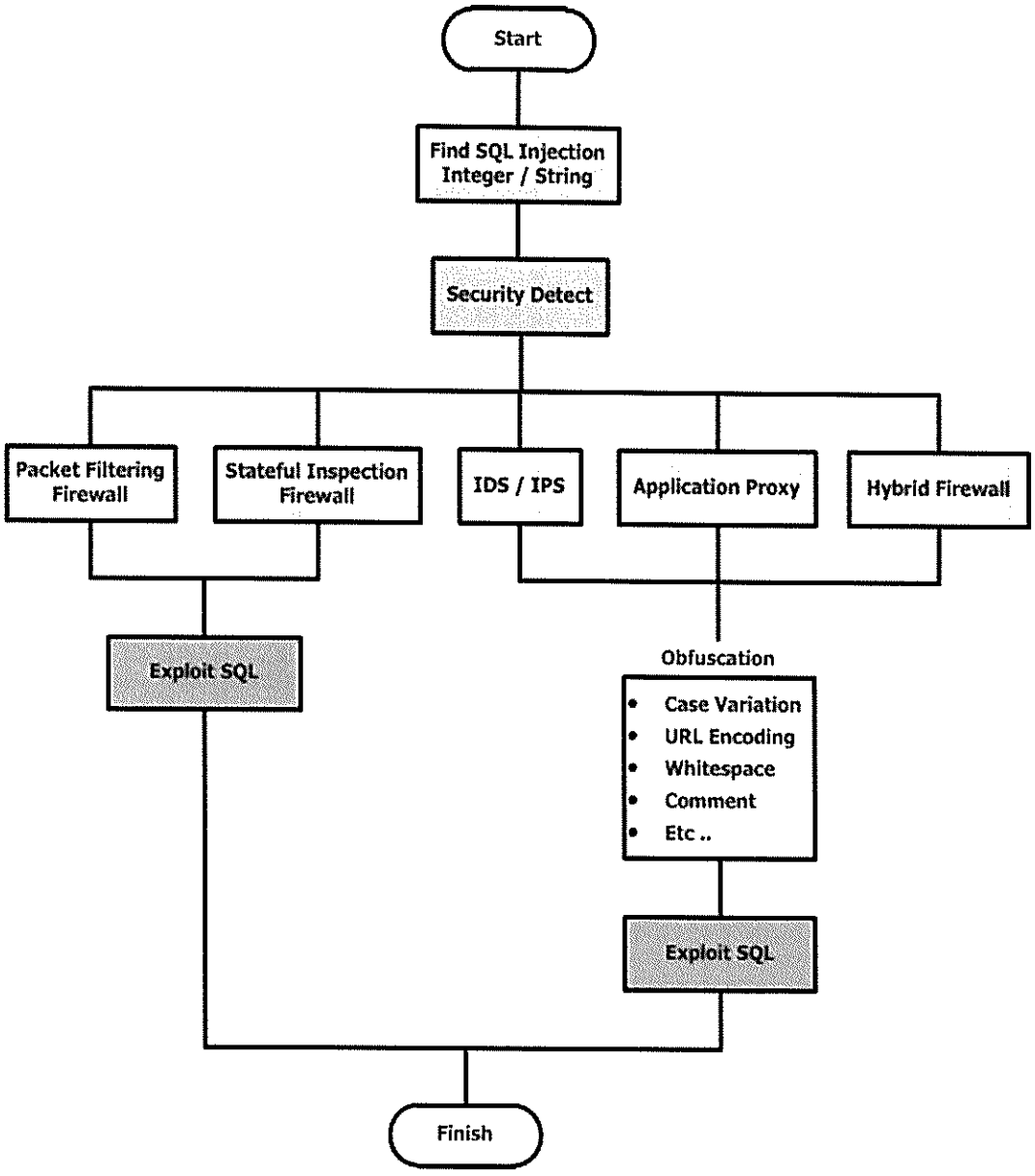
รูปแบบการติดตั้งโครงสร้างระบบรักษาความปลอดภัยที่มี Firewall, IDS/IPS และ Application Proxy



ภาพที่ 2.3 โครงสร้างระบบรักษาความปลอดภัยที่มี Firewall, IDS/IPS และ Application Proxy

2.3 ออกแบบการหลบหลีกจากการตรวจจับของระบบรักษาความปลอดภัย

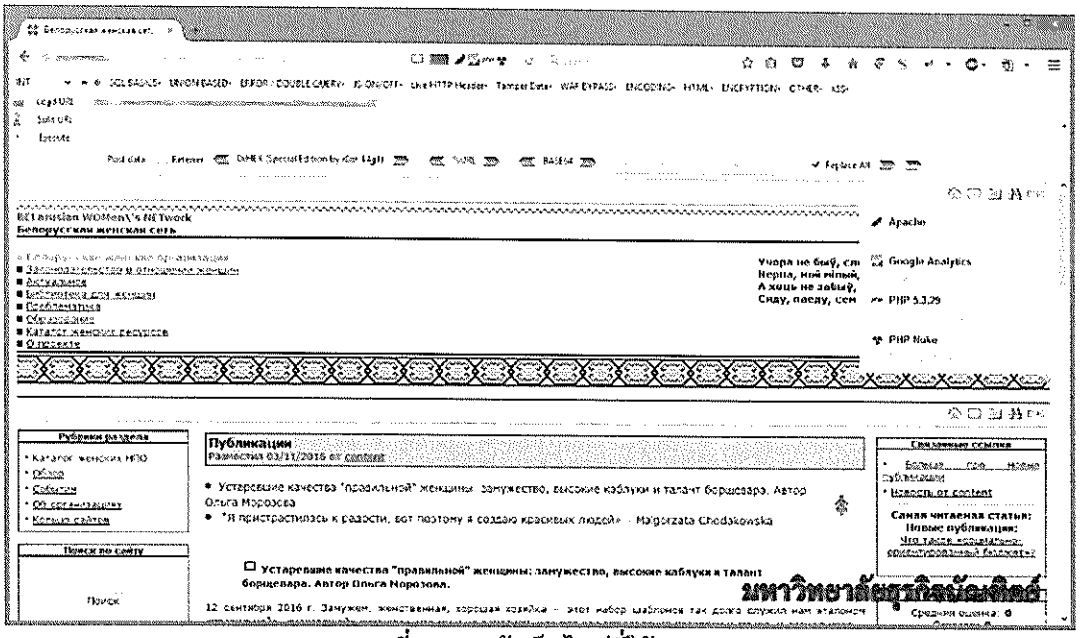
ผู้วิจัยได้ทำการวิเคราะห์ความสามารถในการตรวจจับและป้องกันภัยคุกคามที่เกี่ยวข้องกับระบบเว็บแอพพลิเคชั่นของอุปกรณ์รักษาความปลอดภัยแต่ละประเภทแล้วพบว่า อุปกรณ์ Packet Filtering Firewall, Stateful Inspection Firewall นั้นไม่มีความสามารถในการตรวจจับและป้องกันการโจมตีเว็บแอพพลิเคชั่น แต่ในส่วนของอุปกรณ์ IDS/IPS, Application Proxy และ Hybrid Firewall ในบางระบบนั้นก็มีความสามารถที่จะตรวจจับและป้องกันการโจมตีเว็บแอพพลิเคชั่นด้วยเทคนิค SQL Injection ได้บ้าง เนื่องจากอุปกรณ์ IDS/IPS, Application Proxy และ Hybrid Firewall นั้นจะมี Signature สำหรับการตรวจจับและป้องกันการโจมตีด้วย SQL Injection แบบไม่ซับซ้อนได้อยู่บ้าง ตัวอย่างเช่น การส่งค่าอย่าง OR 1=1 และ OR 1=0 เพื่อหาช่องโหว่ SQL Injection นั้นอุปกรณ์ IDS/IPS, Application Proxy และ Hybrid Firewall ส่วนใหญ่จะรู้จักว่าค่าที่ถูกส่งผ่านไปยังเว็บแอพพลิเคชั่นนั้นตรงกันกับ Signature ที่มีอยู่ก็จะทำการหยุดหรือละทิ้งแพ็คเก็จข้อมูลดังกล่าว แต่หากว่าเปลี่ยนการส่งค่าให้มีลักษณะที่ซับซ้อนขึ้นเพื่อที่จะพยายามหลบหลีก (Obfuscation) การตรวจจับและป้องกันอาจทดลองส่งค่าอย่าง oR/**/5-3=2 และ oR/**/5-3=0 ซึ่งการป้องกันในบางระบบจะทำการปล่อยผ่านแพ็คเก็จข้อมูลดังกล่าว เนื่องจากตรงกันกับ Signature ที่มีอยู่



ภาพที่ 2.4 การหลบหลีกจากการตรวจจับของระบบรักษาความปลอดภัย

2.4 ทดสอบเจาะระบบเว็บไซต์ด้วยเทคนิค SQL Injection ผ่าน HTTP Header

ผู้วิจัยได้ทำการสุ่มทดสอบเจาะระบบเว็บไซต์ด้วยเทคนิค SQL Injection ผ่าน HTTP Header ซึ่งใช้ขั้นตอนในการทดสอบดังต่อไปนี้

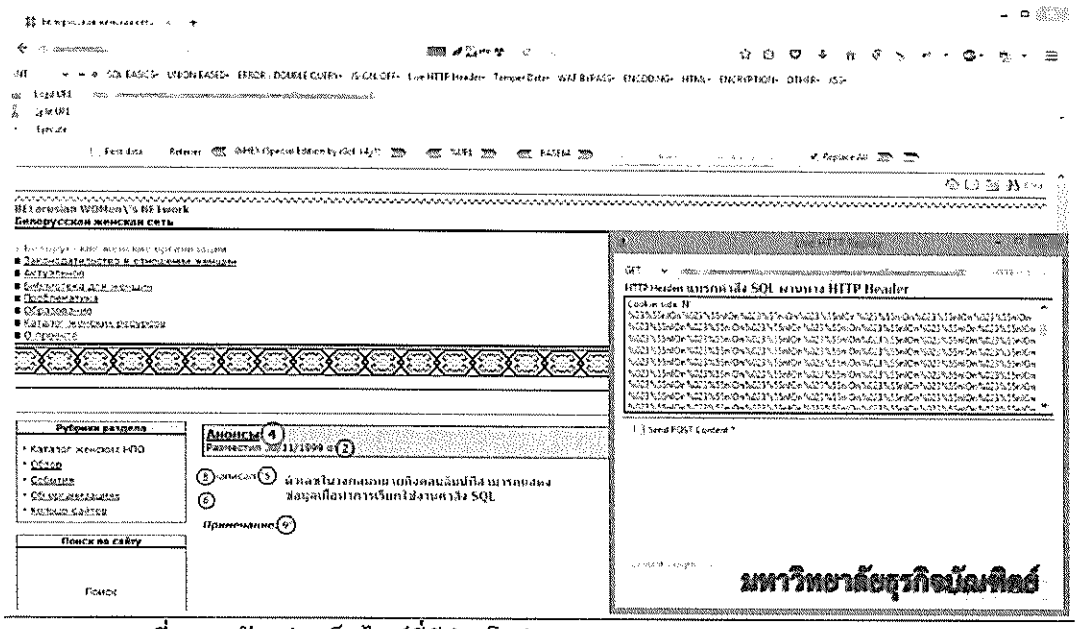


ภาพที่ 2.5 หน้าเว็บไซต์ที่ใช้ทดสอบ

ผู้วิจัยพบว่าในเว็บไซต์ดังกล่าวนี้มีช่องโหว่ SQL Injection แต่ในการส่งการร้องขอผ่าน HTTP Method แบบ GET ก็ตามที่มีคำสั่งของ SQL แล้วจำเป็นต้องเรียกใช้งานตัวอักขระอย่างเครื่องหมาย () เช่นการใช้งานคำสั่งอย่างเช่น version(), user() หรือ database() ก็ตามนั้น เว็บแอพพลิเคชั่นนี้จะแสดงข้อความตอบกลับมาทันทีว่า "The html tags you attempted to use are not allowed"

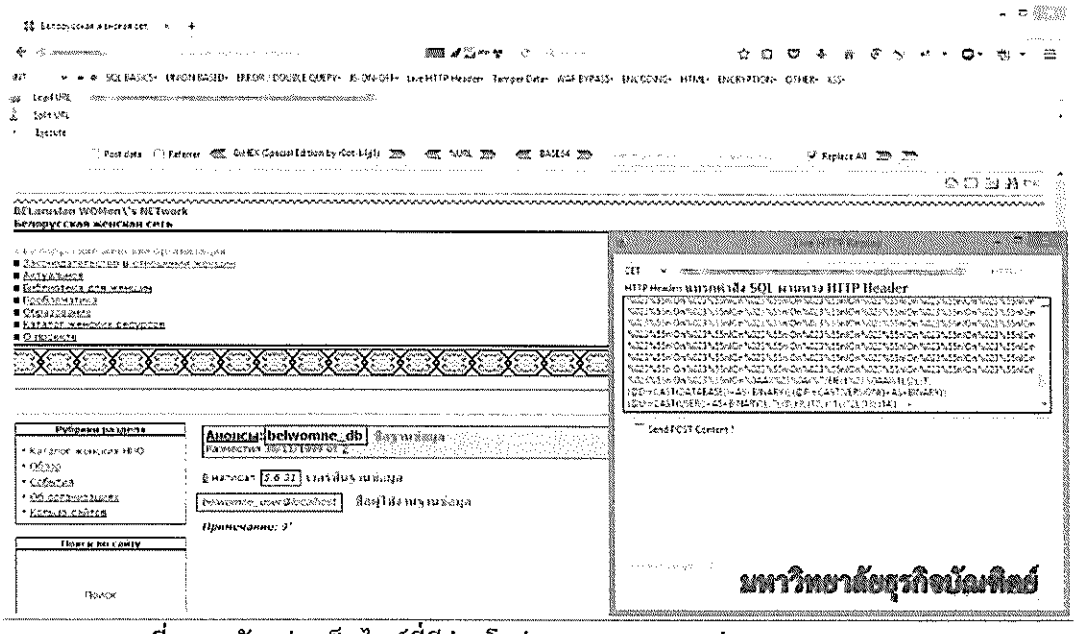
จากนั้นผู้วิจัยได้ทำการสันนิษฐานในการทดสอบว่า หากเปลี่ยนแปลงข้อมูลในส่วนพารามิเตอร์ที่ใช้ในการร้องขอผ่าน HTTP Method นั้น โดยให้พารามิเตอร์ที่ใช้ในการร้องขออยู่ใน HTTP Header แทนซึ่งผลลัพธ์ที่ได้นั้น ผู้วิจัยสามารถที่จะส่งการร้องขอเรียกใช้งานในรูปแบบของโจมตีด้วย SQL Injection ผ่าน HTTP Header ในครั้งเดียวแล้วได้ข้อมูลที่อยู่ในฐานข้อมูลออกมา

ทำการตรวจสอบหาช่องโหว่ SQL Injection ผ่านทาง HTTP Header



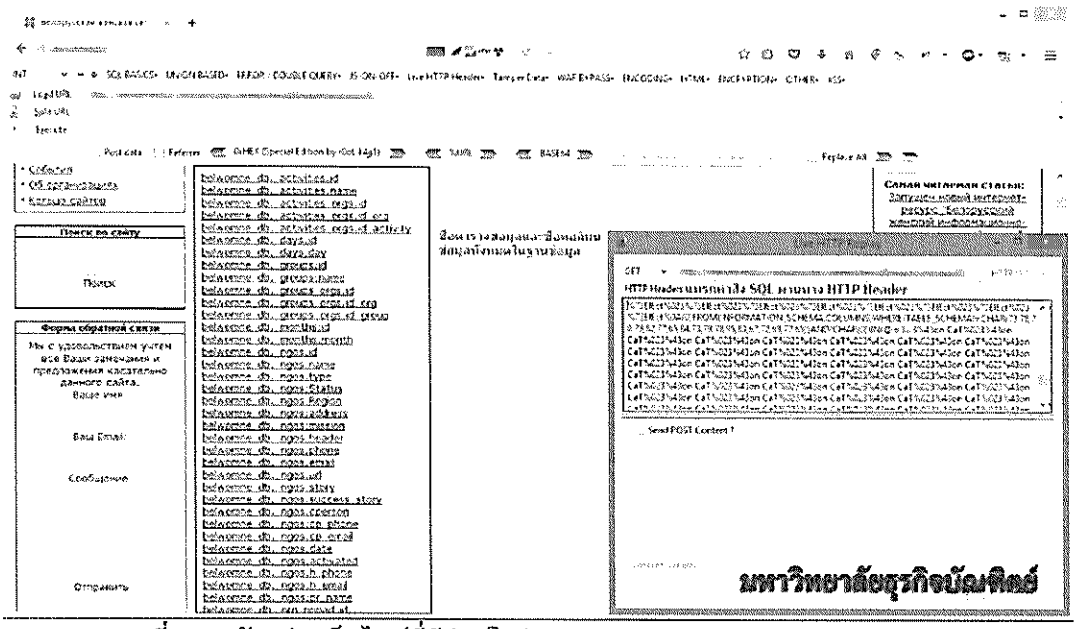
ภาพที่ 2.6 ตัวอย่างเว็บไซต์ที่มีช่องโหว่ SQL Injection ผ่าน HTTP Header (1)

ทำการแทรกคำสั่ง SQL ที่ชื่อฐานข้อมูล, แสดงเวอร์ชั่นฐานข้อมูล, ชื่อผู้ใช้งานฐานข้อมูล ผ่านทาง HTTP Header



ภาพที่ 2.7 ตัวอย่างเว็บไซต์ที่มีช่องโหว่ SQL Injection ผ่าน HTTP Header (2)

ทำการแทรกคำสั่ง SQL ที่ดึงข้อมูลต่าง ๆ ผ่านทาง HTTP Header



ภาพที่ 2.8 ตัวอย่างเว็บไซต์ที่มีช่องโหว่ SQL Injection ผ่าน HTTP Header (3)

ซึ่งผู้วิจัยยังพบว่าเว็บไซต์อีกเป็นจำนวนไม่น้อยที่สามารถเข้าถึงได้โดยผ่านระบบอินเตอร์เน็ต และทำให้สามารถถูกเจาะระบบด้วยเทคนิคดังกล่าวนี้ได้

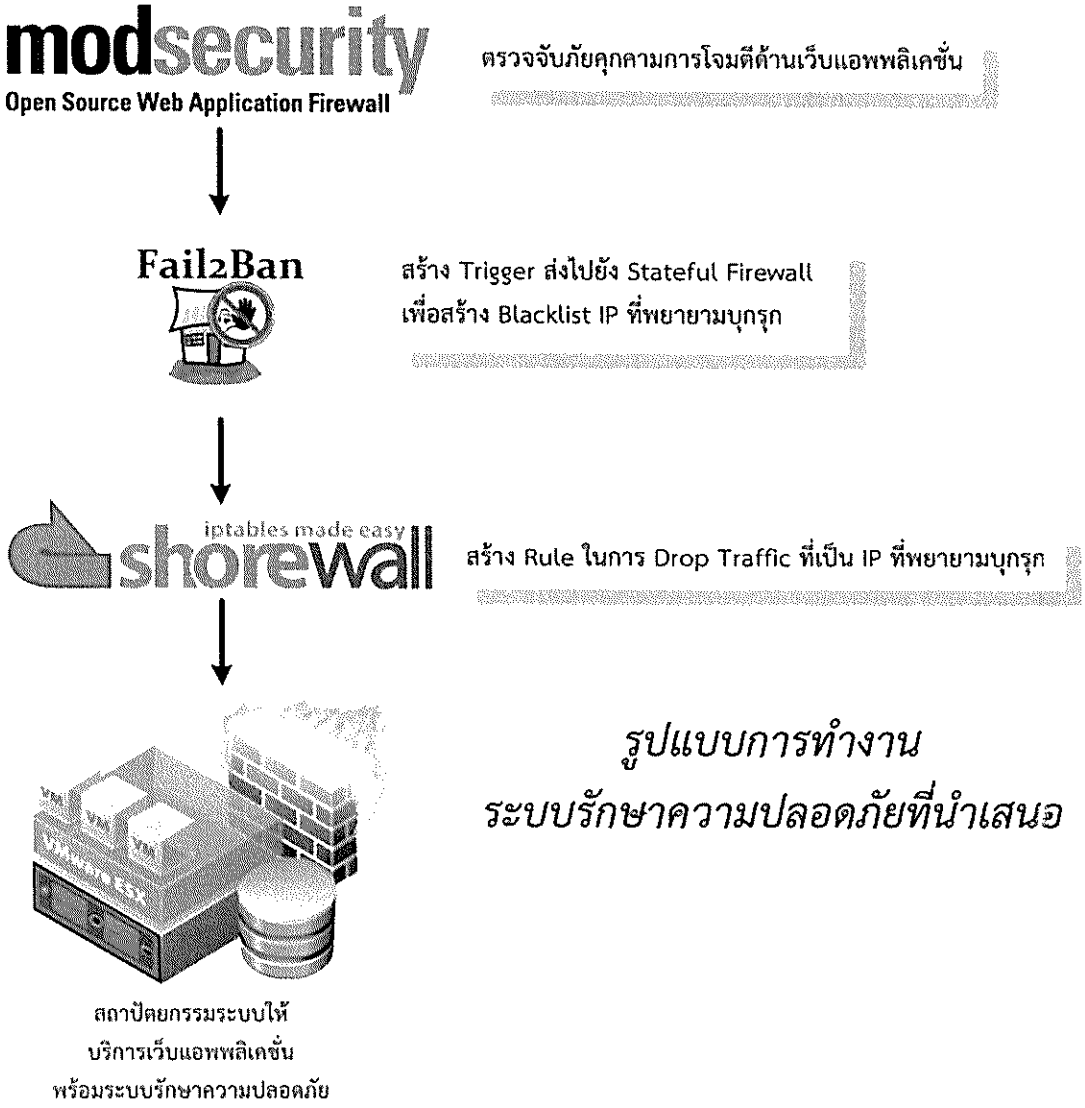
ผลการศึกษา

จากการทดลองในงานวิจัยที่ได้ทำการศึกษานั้น พบว่าเว็บไซต์ขององค์กรหรือหน่วยงานต่าง ๆ ที่ได้ทำการทดสอบเจาะระบบเว็บไซต์นั้น มีเว็บไซต์จำนวนไม่น้อยที่มีช่องโหว่ SQL Injection ซึ่งผู้วิจัยคาดการณ์ว่าผู้ที่มีหน้าที่ดูแลรับผิดชอบระบบเหล่านั้น อาจจะยังไม่ทราบถึงปัญหาหรือภัยคุกคามของระบบเหล่านั้น จึงยังทำให้เกิดข้อผิดพลาดอยู่ในระบบหรืออาจจะทราบแล้วแต่ยังไม่สามารถหาแนวทางแก้ไขหรือป้องกันระบบให้ลดความเสี่ยงจากภัยคุกคามระบบคอมพิวเตอร์นั้นได้

จากการทดลองดังกล่าวนั้น ผู้วิจัยพบว่าแม้ระบบ Web Application Firewall จะสามารถป้องกันการคุกคามจากการ SQL Injection ได้นั้น แต่การปฏิเสธการโจมตีโดยไม่ตอบสนองค่าข้อมูล แต่ตอบสนองในรูปแบบ Message หรือ HTTP Status ต่าง ๆ นั้นยังเป็นการเปิดโอกาสให้ผู้วิจัยได้ทดลองเจาะระบบด้วย SQL Injection ด้วยการพยายามปรับเปลี่ยนรูปแบบการคุกคามไปเรื่อย ๆ จนประสบความสำเร็จ

ซึ่งการแก้ไขปัญหาช่องโหว่ดังกล่าวนี้ ลำดับถัดมานั้นผู้วิจัยได้เลือกทำการแก้ไขโครงสร้างที่ให้บริการพื้นฐาน (Infrastructure) ของเครื่องเซิร์ฟเวอร์ด้วยการติดตั้งโปรแกรมที่มีชื่อว่า Shorewall ซึ่งโปรแกรมนี้มีหน้าที่ช่วยบริหารจัดการโปรแกรม Stateful Firewall ตัวอย่างเช่น iptables ที่มีอยู่บน

ระบบปฏิบัติการแบบ Linux และติดตั้งโปรแกรมที่มีชื่อว่า Fail2Ban ซึ่งผู้วิจัยได้ทำการออกแบบระบบให้มีการป้องกันเว็บเซิร์ฟเวอร์ให้ทำงานร่วมกันกับ Stateful Firewall โดยให้โปรแกรม Fail2Ban ทำหน้าที่เป็น Trigger ในการตรวจจับความถี่ของการคุกคามที่ไม่ประสบความสำเร็จ และบอกให้ Stateful Firewall Drop Packet ทิ้ง



ภาพที่ 3.1 แสดงรูปแบบระบบรักษาความปลอดภัยที่นำเสนอ

เมื่อ Web Application Firewall นั้นตรวจพบความพยายามที่จะโจมตีด้วย SQL Injection เข้ามานั้น โปรแกรม Fail2Ban ก็ทำการสร้าง Trigger ไปบอกโปรแกรม Shorewall เพื่อที่จะให้สร้าง rule สำหรับระบุ IP ที่เป็น Blacklist บน Stateful Firewall