Module 14
SQL Injection

Các Chủ Đề Chính Trong Chương Này

Tấn Công SQl Injection Là Gì

Các Lỗi SQL Thường Gặp

Một Số Dạng Tấn Công SQL Injection Thông Dụng

Phòng Chống Tấn Công SQL Injection

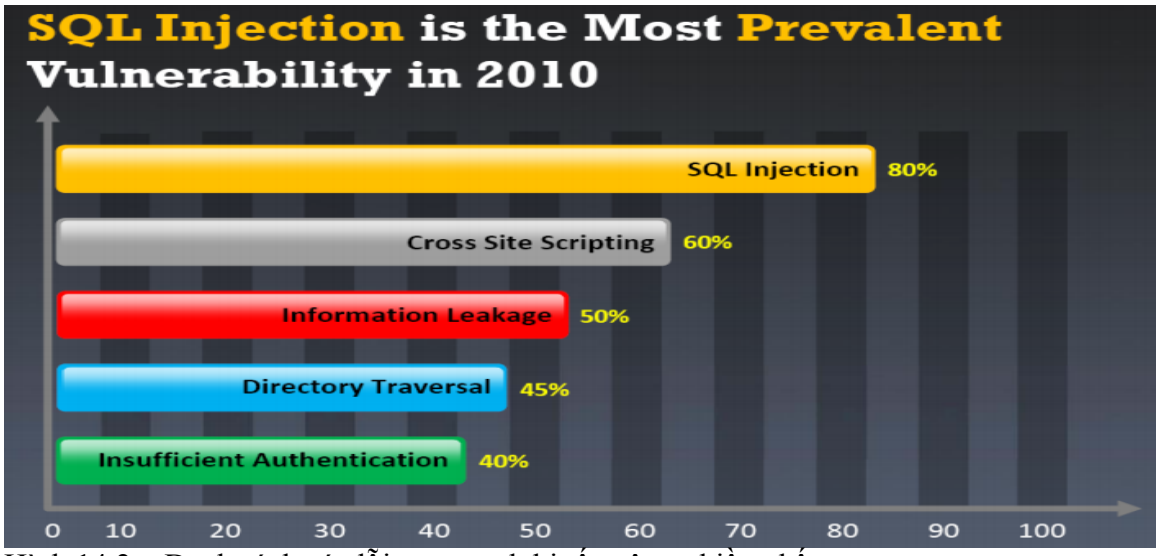
Giới Thiệu Về SQL Injection

Đa số ứng dụng web ngày này đều quản lý và đáp ứng các yêu cầu truy xuất dữ liệu thông qua ngôn ngữ truy vấn cấu trúc SQL. Các hệ quản trị cơ sở dữ liệu thông dụng như Oracle, MS SQL hay MySQL đều có chung một đặc điểm này, chính vì vậy những dạng tấn công liên quan đến SQL thường được xếp hàng đầu trong danh sách các lỗ hổng nguy hiểm nhất, và dạng tấn công vào những lỗi này gọi là SQL injection.

Vào tháng 12 năm 2010 một đợt tấn công SQl injection đã lấy đi hàng trăm ngàn thông tin khách hàng, hoặc tin về hacker *Albert Gonzalez* được cho là đã lấy cắp 130 triệu thông tin thẻ tín dụng thông qua tấn công SQL injection vào các website và cài đặt công cụ sniffer để đánh cắp dữ liệu, đây được xem là một đợt trộm cắp thông tin người dùng được lớn nhất trong lịch sử Hoa Kỳ được đăng trên InformationWeek trong Hình 14.1.

Hình 14.1 – Thông tin các thẻ tín dụng bị đánh cắp

Chúng ta thấy, các tấn công dạng này của hacker thường đánh vào các trang web chứa thông tin tài khoản quan trọng của người dùng trên các trang web thương mại điện tử, kết quả thu được có giá trị kinh tế cao, và khả năng thành công lớn, dễ tiến hành là những đặc điểm khiến cho SQL injection được xếp hàng số 1 trong danh sách những lỗi bị ảnh hưởng nhiều nhất trong năm 2010 như Hình 14.2



Hình 14.2 – Danh sách các lỗi trang web bị tấn công nhiều nhất

Vậy SQL injection là gì và các hacker thực hiện chúng như thế nào mà nguy hiểm đến vậy, các bạn hãy xem phần mô tả tương đối đầy đủ về dạng tấn công này qua trang wikipedia.org như sau :

> "SQL injection là một kỹ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng của việc kiểm tra dữ liệu đầu vào trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu trả về để inject (tiêm vào) và thi hành các câu lệnh SQL bất hợp pháp, Sql injection có thể cho phép những kẻ tấn công thực hiện các thao tác, delete, insert, update,… trên cơ sỡ dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy, lỗi này thường xãy ra trên các ứng dụng web có dữ liệu được quản lý bằng các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, Oracle, DB2, Sysbase...

Các Lỗi Thường Gặp

Không kiểm tra ký tự thoát truy vấn

Đây là dạng lỗi SQL injection xảy ra khi thiếu đoạn mã kiểm tra dữ liệu đầu vào trong câu truy vấn SQL. Kết quả là người dùng cuối có thể thực hiện một số truy vấn không mong muốn đối với cơ sở dữ liệu của ứng dụng. Dòng mã sau sẽ minh họa lỗi này:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

Câu lệnh này được thiết kế để trả về các bản ghi tên người dùng cụ thể từ bảng những người dùng. Tuy nhiên, nếu biến "userName" được nhập chính xác theo một cách nào đó bởi người dùng ác ý, nó có thể trở thành một câu truy vấn SQL với mục đích khác hẳn so với mong muốn của tác giả đoạn mã trên. Ví dụ, ta nhập vào giá trị của biến **userName** như sau:

```
a' or 't'='t
```

Khiến câu truy vấn có thể được hiểu như sau:

```
SELECT * FROM users WHERE name = 'a' OR 't'='t';
```

Nếu đoạn mã trên được sử dụng trong một thủ tục xác thực thì ví dụ trên có thể được sử dụng để bắt buộc lựa chọn một tên người dùng hợp lệ bởi 't'='t' luôn đúng. Trong khi hầu hết các SQL server cho phép thực hiện nhiều truy vấn cùng lúc chỉ với một lần gọi, tuy nhiên một số SQL API như mysql_query của php lại không cho phép điều đó vì lý do bảo mật. Điều này chỉ ngăn cản tin tặc tấn công bằng cách sử dụng các câu lệnh riêng rẽ mà không ngăn cản tin tặc thay đổi các từ trong cú pháp truy vấn. Các giá trị của biến "userName" trong câu truy vấn dưới đây sẽ gây ra việc xoá những người dùng từ bảng người dùng cũng tương tự như việc xóa tất cả các dữ liệu được từ bảng dữ liệu (về bản chất là tiết lộ các thông tin của mọi người dùng), ví dụ này minh họa bằng một API cho phéo thực hiện nhiều truy vấn cùng lúc:

```
a';DROP TABLE users; SELECT * FROM data WHERE 't' = 't
```

Khiến câu truy vấn có thể được hiểu như sau:

```
SELECT * FROM users WHERE name = 'a' OR 't'='t';
```

Nếu đoạn mã trên được sử dụng trong một thủ tục xác thực thì ví dụ trên có thể được sử dụng để bắt buộc lựa chọn một tên người dùng hợp lệ bởi 't'='t' luôn đúng. Trong khi hầu hết các SQL server cho phép thực hiện nhiều truy vấn cùng lúc chỉ với một lần gọi, tuy nhiên một số SQL API như mysql_query của php lại không cho phép điều đó vì lý do bảo mật. Điều này chỉ ngăn cản tin tặc tấn công bằng cách sử dụng các câu lệnh riêng rẽ mà không ngăn cản tin tặc thay đổi các từ trong cú pháp truy vấn. Các giá trị của biến "userName" trong câu truy vấn dưới đây sẽ gây ra việc xoá những người dùng từ bảng người dùng cũng tương tự như việc xóa tất cả các dữ liệu được từ bảng dữ liệu (về bản chất là tiết lộ các thông tin của mọi người dùng), ví dụ này minh họa bằng một API cho phéo thực hiện nhiều truy vấn cùng lúc:

```
a';DROP TABLE users; SELECT * FROM data WHERE 't' = 't
```

Điều này đưa tới cú pháp cuối cùng của câu truy vấn trên như sau:

```
SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT * FROM
DATA WHERE 't' = 't';
```

Xử lý không đúng kiểu

Lỗi SQL injection dạng này thường xảy ra do lập trình viên hay người dùng định nghĩa đầu vào dữ liệu không rõ ràng hoặc thiếu bước kiểm tra và lọc kiểu dữ liệu đầu vào. Điều

này có thể xảy ra khi một trường số được sử dụng trong truy vấn SQL nhưng lập trình viên lại thiếu bước kiểm tra dữ liệu đầu vào để xác minh kiểu của dữ liệu mà người dùng nhập vào có phải là số hay không. Ví dụ như sau:

```
statement := "SELECT * FROM data WHERE id = " + a_variable + ";"
```

Ta có thể nhận thấy một cách rõ ràng ý định của tác giả đoạn mã trên là nhập vào một số tương ứng với trường id - trường số. Tuy nhiên, người dùng cuối, thay vì nhập vào một số, họ có thể nhập vào một chuỗi ký tự, và do vậy có thể trở thành một câu truy vấn SQL hoàn chỉnh mới mà bỏ qua ký tự thoát. Ví dụ, ta thiết lập giá trị của biến a_variable là:

```
1;DROP TABLE users
```

khi đó, nó sẽ thực hiện thao tác xóa người dùng có id tương ứng khỏi cơ sở dữ liệu, vì câu truy vấn hoàn chỉnh đã được hiểu là:

```
SELECT * FROM DATA WHERE id=1;DROP TABLE users;
```

Blind SQL injection

SQL injection dạng này là dạng lỗi tồn tại ngay trong ứng dụng web nhưng hậu quả của chúng lại không hiển thị trực quan cho những kẻ tấn công. Nó có thể gây ra sự sai khác khi hiển thị nội dung của một trang chứa lỗi bảo mật này, hậu quả của sự tấn công SQL injection dạng này khiến cho lập trình viên hay người dùng phải mất rất nhiều thời gian để phục hồi chính xác từng bit dữ liệu. Những kẻ tấn công còn có thể sử dụng một số công cụ để dò tìm lỗi dạng này và tấn công với những thông tin đã được thiết lập sẵn.

Một Số Dạng Tấn Công SQL Injection Thông Dụng

Có bốn dạng tấn công thường gặp bao gồm: vượt qua kiểm tra lúc đăng nhập, sử dụng câu lệnh SELECT, sử dụng câu lệnh INSERT, sử dụng các stored-procedures.

Dạng tấn công vượt qua kiểm tra lúc đăng nhập

Với dạng tấn công này, tin tặc có thể dễ dàng vượt qua các trang đăng nhập nhờ vào lỗi khi dùng các câu lệnh SQL thao tác trên cơ sở dữ liệu của ứng dụng web. Thông thường để cho phép người dùng truy cập vào các trang web được bảo mật, hệ thống thường xây dựng trang đăng nhập để yêu cầu người dùng nhập thông tin về tên đăng nhập và mật khẩu. Sau khi người dùng nhập thông tin vào, hệ thống sẽ kiểm tra tên đăng nhập và mật khẩu có hợp lệ hay không để quyết định cho phép hay từ chối thực hiện tiếp.

Dạng tấn công sử dụng câu lệnh SELECT

Dạng tấn công này phức tạp hơn. Để thực hiện được kiểu tấn công này, kẻ tấn công phải có khả năng hiểu và lợi dụng các sơ hở trong các thông báo lỗi từ hệ thống để dò tìm các điểm yếu khởi đầu cho việc tấn công. Ví dụ, trong các trang tìm kiếm. Các trang này cho phép người dùng nhập vào các thông tin tìm kiếm như Họ, Tên, … Đoạn mã thường gặp là:

```
    <%
    Dim vAuthorName, objRS, strSQL
    vAuthorName = Request("fAUTHOR_NAME")
    strSQL = "SELECT * FROM T_AUTHORS WHERE AUTHOR_NAME =' " & _
vAuthorName & " ' "
    Set objRS = Server.CreateObject("ADODB.Recordset")
    objRS.Open strSQL, "DSN=..."
    …
    Set objRS = Nothing %>
```

Tương tự như trên, tin tặc có thể lợi dụng sơ hở trong câu truy vấn SQL để nhập vào trường tên tác giả bằng chuỗi giá trị:

```
' UNION SELECT ALL SELECT OtherField FROM OtherTable WHERE ' '=' (*)
```

Lúc này, ngoài câu truy vấn đầu không thành công, chương trình sẽ thực hiện thêm lệnh tiếp theo sau từ khóa UNION nữa. Giả sử đoạn mã nhập vào là:

```
' DROP TABLE T_AUTHORS --
```

Câu truy vấn sẽ thực hiện việc xóa bảng.

Dạng tấn công sử dụng câu lệnh INSERT

Thông thường các ứng dụng web cho phép người dùng đăng kí một tài khoản để tham gia. Chức năng không thể thiếu là sau khi đăng kí thành công, người dùng có thể xem và hiệu chỉnh thông tin của mình. SQL injection có thể được dùng khi hệ thống không kiểm tra tính hợp lệ của thông tin nhập vào. Ví dụ, một câu lệnh INSERT có thể có cú pháp dạng:

```
    INSERT INTO TableName VALUES('Value One', 'Value Two', 'Value
Three')
```

Nếu đoạn mã xây dựng câu lệnh SQL có dạng :

```
<%
```

```
    strSQL = "INSERT INTO TableName VALUES(' " & strValueOne & " ', ' "
_ & strValueTwo & " ', ' " & strValueThree & " ') "
    Set objRS = Server.CreateObject("ADODB.Recordset")
    objRS.Open strSQL, "DSN=..."
    …
    Set objRS = Nothing %>
```

Thì chắc chắn sẽ bị lỗi SQLi, bởi vì nếu ta nhập vào trường thứ nhất ví dụ như:

```
    ' + (SELECT TOP 1 FieldName FROM TableName) + '
```

Lúc này câu truy vấn sẽ là :

```
    INSERT INTO TableName VALUES(' ' + (SELECT TOP 1 FieldName FROM
TableName) + ' ', 'abc', 'def')
```

Khi đó, lúc thực hiện lệnh xem thông tin, xem như bạn đã yêu cầu thực hiện thêm một lệnh nữa đó là:

```
    SELECT TOP 1 FieldName FROM TableName
```

Dạng tấn công sử dụng stored-procedures

Việc tấn công bằng stored-procedures sẽ gây tác hại rất lớn nếu ứng dụng được thực thi với quyền quản trị hệ thống 'sa'. Ví dụ, nếu ta thay đoạn mã tiêm vào dạng: ' ; EXEC xp_cmdshell ‘cmdd.exe dir C: '. Lúc này hệ thống sẽ thực hiện lệnh liệt kê thư mục trên ổ đĩa C:\ cài đặt server. Việc phá hoại kiểu nào tuỳ thuộc vào câu lệnh đằng sau cmd.exe. fg

“

Qua các mô tả khá đầy đủ trên các bạn đã có môt cái nhìn tổng quan về SQL injection, để tấn công dạng này các hacker thường tiến hành các bước như sau :

1. Sư dụng trình duyệt tìm kiếm các trang đăng nhập của web site hay những trang tiếp nhận dữ liệu đầu vào như các form khai báo mất và phục hồi mật khẩu. Kiểm tra các trang hiển thị lệnh POST hay GET bằng cách tìm kiếm trong source code như trong bài tập hack trên trang web học tấn công khá nổi tiếng là www.hackthisite.org

2. Tiếp theo, kiểm tra máy chủ SQL bằng các truy vấn dùng cặp dấu nháy đơn (‘’) nếu máy chủ đáp ứng bằng các thông điệp như use ‘a’=’a’ hay tương tự như vậy thì các hacker có thể đáon được máy chủ có khả năng bị tấn công theo dạng SQL injection. Ngoài ra, các dữ liệu thường dùng cho quá trình kiểm tra lỗi khác như ‘ or 1=1– hay ” or 1=1–

3. Khi phát hiện thấy máy chủ hay trang web bị lỗi hacker sẽ dùng các lệnh SELECT để lấy dữ liệu từ máy chủ hay lệnh INSERT đến chèn thông tin vào cơ sở dữ liệu.

Hình 14.3 – Attacker khai thác lỗi SQL injection

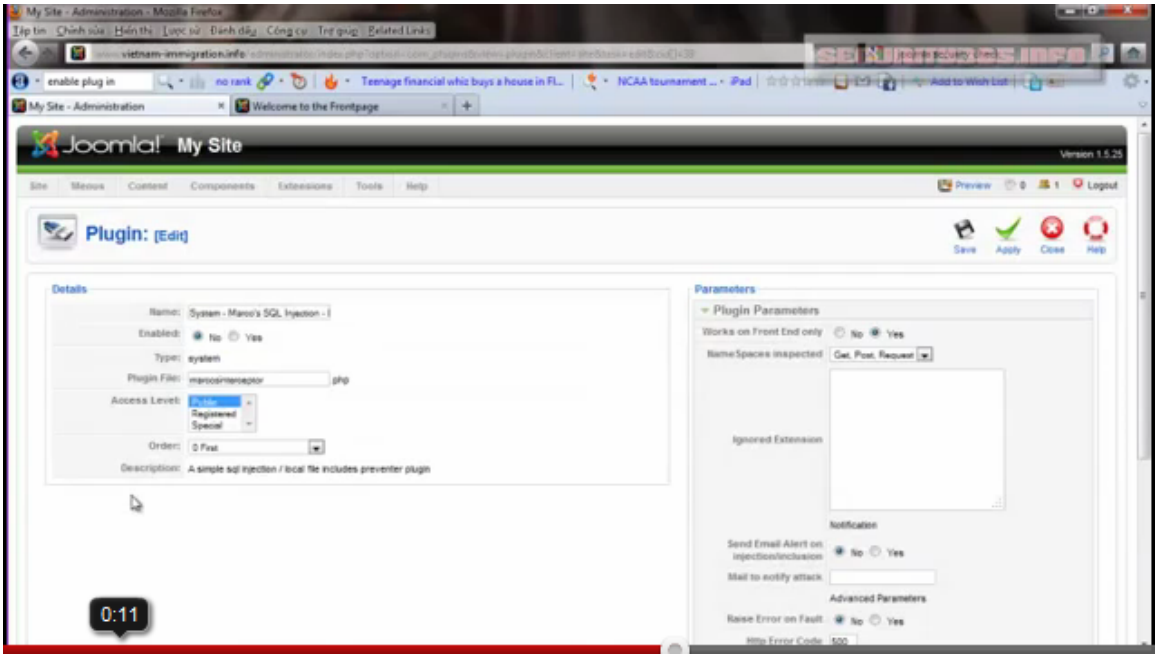
Các bước truy vân dữ liệu và khai thác trên thường được tự động hóa với các công cụ tấn công chuyên dùng cho SQL injection như Havji, SQL Ninja. Còn quá trình kiểm tra thì sử dụng Web Acunetix Scanner, W3AF …Sau đây là danh sách các công cụ chuên dùng để tấn công SQL injection hiện nay :

Hình 14.4 – Các công cụ tấn công SQL injection

Phòng Chống Tấn Công SQL Injection

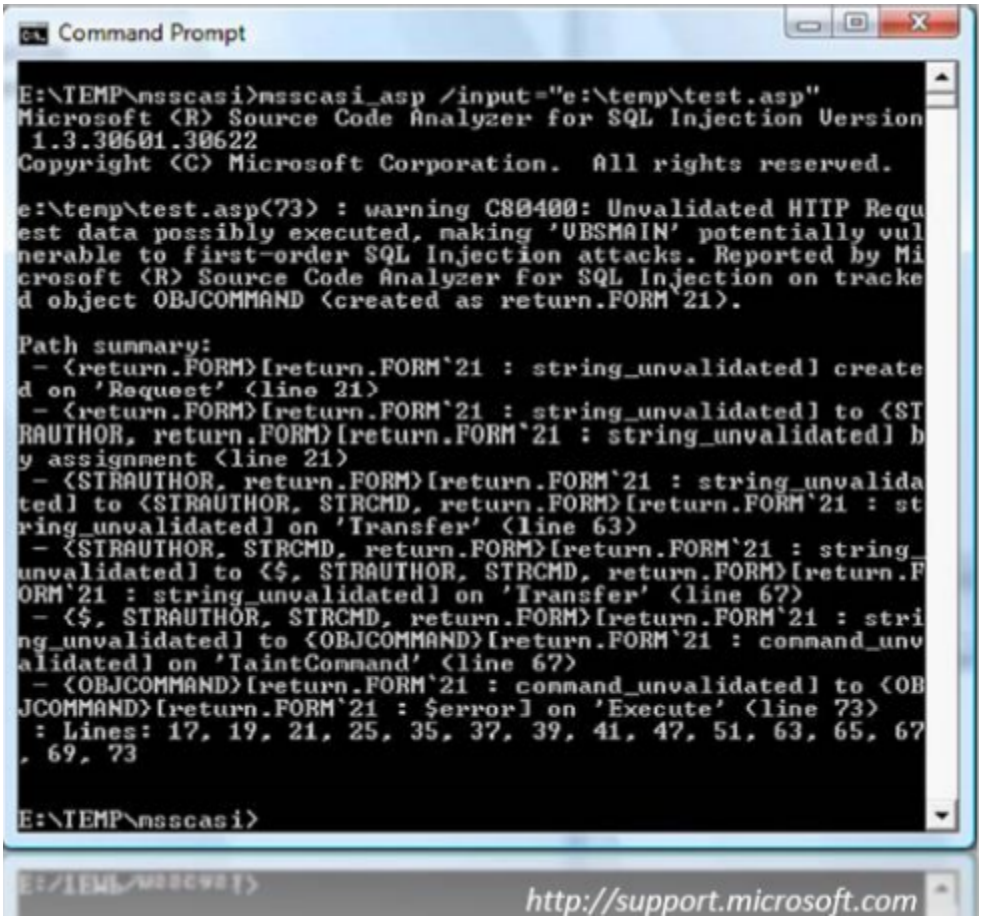
Để phòng chống tấn công SQL Injection trước hết chúng ta cần gán quyền thích hợp cho người dùng khi kết nối đến cơ sở dữ liệu và áp dụng mật khẩu mạnh cho tài khoản SA hay tài khoản quản trị. Các hacker thường chẩn đoán lỗi dựa trên các thông báo trả về, do đó việc hạn chế các thông báo này cũng giúp tăng cường an ninh cho website hay máy chủ web và giảm thiểu rũi ro bị tấn công SQL injection.

Ngoài ra, các kí tự mà hacker nhập vào để gởi đến máy chủ thường có các kí tự đặc biệt như dấu nháy đơn (‘ ’) nên các bạn cần kiểm tra chặt chẽ các dữ liệu nhập vào của người dùng trong mã nguồn, từ chối các truy vấn khả nghi hay cài đặt các module, plug-in hỗ trợ phòng chống SQL Injection. Ví dụ như khi dùng mã nguồn mở Joomla để xây dựng trang web thì nên cài đặt plug-in Anti SQL, LFI cho trang web như minh hoạt tại đây : http://www.youtube.com/watch?v=LdXrHurs-_o hay thường xuyên kiểm tra hệ thống bằng các chương trình quét lỗi website trực tuyến như minh họa tại http://www.youtube.com/watch?v=_l7P1_uZexc



Hình 14.5 - Hình minh họa sau khi cài đặt plug-in anti sqli & lfi

Việc dò tìm mã nguồn xem có bị SQL Injection hay không ta có thể dùng Microsoft Source Code Analyzer hay Microsft URL Scan



Hình 14.6 - Kết quả kiểm tra một tập tin với Microsoft Source Code Analyzer

Bên cạnh đó, các máy chủ cơ sở dữ liệu cần được cập nhật và vá lỗi đầy đủ, nên đặt sau hệ thống tường lữa để tránh sự tương tác trực tiếp, điều này sẽ giúp tăng cường bảo mật.Trên hệ thống cần có thêm các dịch vụ hay thiết bị dò tìm xâm phạm trai phép (IDS/IPS) như SNORT, Cisco IDS …

Tổng Kết

Qua chương này chúng ta đã biết được lỗi bảo mật SQl injection là gì và những công cụ mà hacker thường sử dụng để tấn công vào các trang web dính lỗi trên. Để phòng chống bị tấn công các bạn cần chú ý kiểm tra các biến hay dữ liệu đầu vào, đặc biệt là những form nhập liệu trên trang web, lọc các kí tự đặc biệt mà hacker có thể dùng để kiểm tra hay gởi lệnh truy vấn nguy hiểm đến cơ sở dữ liệu. Trong chương tiếp theo chúng ta sẽ tìm hiểu về “Tấn công mạng không dây”.