# Chương 1: Tấn công Blind SQL injection.

Lỗ hổng SQL injecton đã tồn tại ngay từ lần đầu tiên ứng dụng web được kết nối với cơ sở dữ liệu. Tuy nhiên, Rain Forest Puppy được coi là người đâu tiên có công bố rộng rãi về lỗ hổng này. Vào ngày giáng sinh năm 1998, Rain Forest Puppy đã viết bài báo “NT Web Technology Vulnerabilities” trên Phrack ([www.phrack.com/issues.html?issue=54&id=8#article](http://www.phrack.com/issues.html?issue=54&id=8#article)), một chuyên trang dành cho các hacker trên thế giới. Tác giả cũng cung cấp một bài hướng dẫn tấn công SQL injection tên “How I hacked PacketStorm” tại www.wiretrip.net/rfp/txt/rfp2k01.txt vào đầu những năm 2000. Kể từ đây, nhiều nhà nghiên cứu trên thế giới đã bắt đầu phát triển các kỹ thuật tấn công, khai thác lỗ hổng SQL injection.

## Lỗ hổng SQL injection.

Trong nhiều năm qua chúng ta đã được nghe nhiều về lỗ hổng SQL injection. Đó thường là những bài viết về kinh nghiệm, cách khai thác lỗ hổng này. SQL injection là một trong những lỗ hổng nguy hiểm nhất, ảnh hưởng nghiêm trọng hoạt động kinh doanh, nghiệp vụ của tổ chức. Nó cho phép kẻ tấn công tiếp xúc với những thông tin cực kỳ quan trọng như: thông tin tài khoản, thông tin email, thông tin thẻ tín dụng.

Trong phần này chúng ta sẽ xem xét nguyên nhân của lỗ hổng. Bắt đầu bằng cái nhìn tổng quan về cấu trúc và hoạt động của ứng dụng web, từ đó cung cấp hiểu biết về cách mà lỗ hổng SQL injection xảy ra dựa trên phân tích mã nguồn của ứng dụng.

### Nguyên tắc hoạt động của ứng dụng web sử dụng cơ sở dữ liệu.

Đại đa số chúng ta sử dụng các ứng dụng web trong cuộc sống hằng ngày, như là một phần trong cuộc sống như: duyệt email, mua hàng trực tuyến, đọc một tin tức mình quan tâm. Ứng dụng web cũng có đủ loại nội dụng, với đủ loại kích thước khác nhau. Mặc dù khác nhau, đa dạng như vậy nhưng các ứng dụng web lại có một điểm giống nhau cơ bản. Đó chính là cách mà ứng dụng web tương tác. Thông thường, đó chính là cơ sở dữ liệu (CSDL).

Ứng dụng web sử dụng cơ sở dữ liệu rất phổ biến ngày nay. Thông thường bao gồm một CSDL phía back-end và một trang web chứa các đoạn mã server-side trên máy chủ, có khả năng trích xuất các dữ liệu từ CSDL dựa trên các hành vi, tác động khác nhau của người dung. Một trong những ví dụ rõ nhất là trang thương mại điện tử: Các thông tin về sản phẩm như: tên, giá, số lượng… được lưu trữ trong CSDL. Tùy thuộc vào loại sản phẩm mà chúng ta muốn xem, ứng dụng sẽ trả về cho chúng ta thông tin sản phẩm tương ứng.

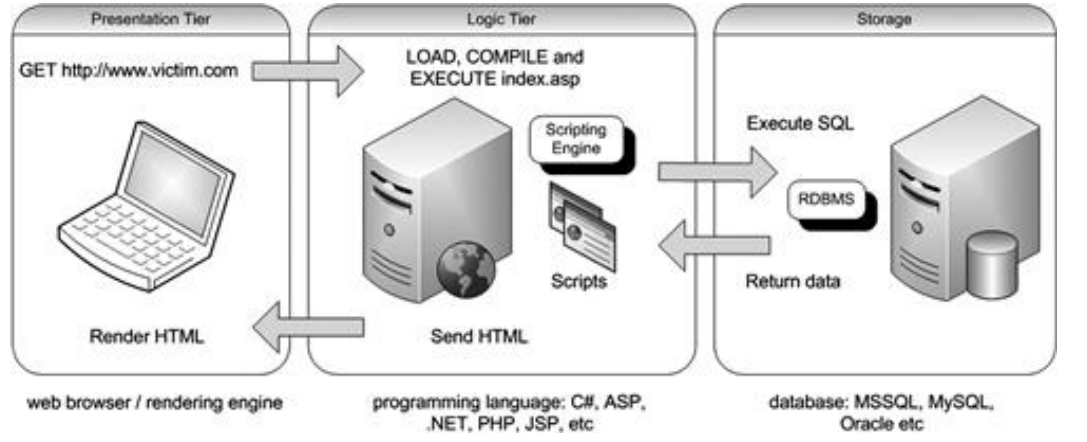
Một ứng dụng web cơ bản gồm 3 lớp (three tiers): Lớp trình bày (Presentation Tier) là trình duyệt, hoặc bộ engine hiển thị, Lớp logic (Logic Tier) là ngôn ngữ lập trình hoạt động phía máy chủ như C#, ASP, .NET, PHP, JSP…, Lớp lưu trữ (Storage Tier) là các hệ quản trị CSDL như Microsoft SQL Server, MySQL, Oracle… Trình duyệt web ở lớp trình bày gửi yêu cầu tới lớp trung gian – lớp logic, tạo ra các yêu cầu truy vấn hoặc cập nhật vào CSDL (lớp lưu trữ).

Trong ví dụ sau, một trang web bán hàng trực tuyến có chức năng cho phép người dùng tìm kiếm/lọc các sản phẩm quan tâm, sắp xếp các sản phẩm, cung cấp tính tùy chọn hiển thị các sản phẩm phù hợp với tài chính, nhu cầu người mua. Để xem tất cả các sản phẩm có giá thấp hơn 100$, người mua sẽ sử dụng URL sau: <http://www.victim.com/products.php?val=100>. Đoạn mã PHP dưới đây sẽ minh họa các mà đầu vào từ người dùng (tham số val) được truyền vào và tạo ra một câu truy vấn SQL động. Đoạn mã sẽ được thực thi khi người dùng gọi tới URL:

|  |
| --- |
| // connect to the database  $conn = mysql\_connect("localhost","username","password");  // dynamically build the sql statement with the input  $query = "SELECT ∗ FROM Products WHERE Price < '$\_GET["val"]' "."ORDER BY ProductDescription";  // execute the query against the database  $result = mysql\_query($query);  // iterate through the record set  while($row = mysql\_fetch\_array($result, MYSQL\_ASSOC))  {  // display the results to the browser  echo "Description : {$row['ProductDescription']} <br>".  "Product ID : {$row['ProductID']} <br>".  "Price : {$row['Price']} <br><br>";  } |

Đoạn code trên minh họa rõ quá trình PHP script tạo ra câu truy vấn. Câu truy vấn sẽ trả về tất cả sản phẩm có giá thấp hơn 100$ trong CSDL, những sản phẩm này sẽ được hiển thị trên trình duyệt web của người dùng. Về nguyên tắc, tất cả các phương pháp làm việc của ứng dụng web đều hoạt động giống như cách này, hoặc một hình thức tương đương nhằm tạo ra câu truy vấn:

|  |
| --- |
| SELECT ∗  FROM Products  WHERE **Price <’100.00’**  ORDER BY ProductDescription; |

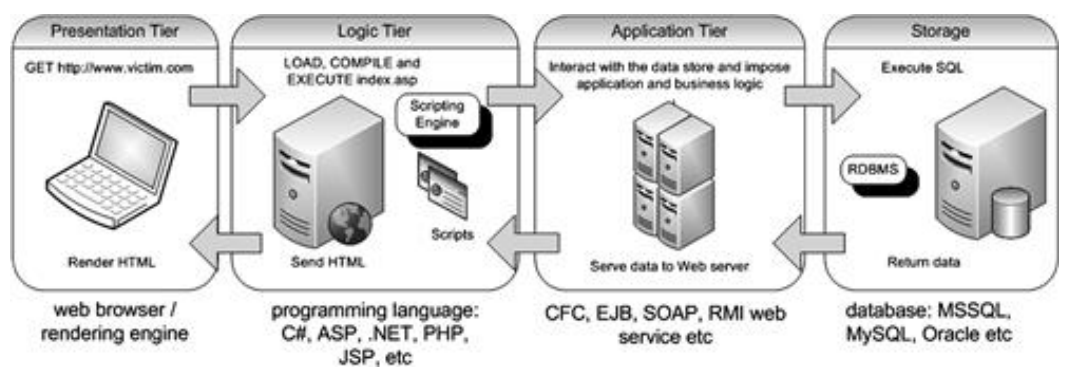


Hình 1: Mô hình ba lớp đơn giản trong ứng dụng web.

Lớp trình bày là lớp cao nhất của ứng dụng, nó hiển thị các thông tin liên quan

các dịch vụ như hàng hóa, nội dung giỏ hàng, và giao tiếp với những lớp khác để trả kết quả lên trình duyệt. Tiếp theo lớp trình bày là lớp logic, thực hiện các chức năng của ứng dụng bằng các xử lý chi tiết. Lớp lưu trữ gồm các máy chủ CSDL, dữ liệu sẽ được lưu trữ và lấy ra tại đây. Điều này giúp cho dữ liệu độc lập với ứng dụng và các hoạt động nghiệp vụ. Một nguyên tắc cơ bản là lớp trình bày không bao giờ liên kết trực tiếp với lớp dữ liệu, mà phải đi qua lớp logic trung gian. Khi người dùng sử dụng trình duyệt web truy cập vào URL: <http://www.victim.com>, máy chủ web server sẽ tải file kịch bản từ file hệ thống, chuyển đến scripting engine để phân tích và thực thi. Đoạn script sẽ thực hiện mở kết nối đến lớp lưu trữ - CSDL qua Database Connector để thực hiện câu truy vấn SQL. CSDL trả về kết quả truy vấn cho Database Connector, chuyển đến cho scripting engine (lớp logic). Lớp logic sẽ thực hiện các ứng dụng, nghiệp vụ tương ứng với kết quả truy vấn SQL trước khi trả về trang Web dạng HTML cho trình duyệt web của người dùng. Trình duyệt web sẽ hiển thị HTML dưới dạng hình ảnh, đồ họa sinh động. Toàn bộ quá trình thực thi chỉ trong khoảng vài giây và hoàn toàn trong suốt với người dùng.

Mô hình ba lớp không có tính mở rộng, do đó trong những năm gần đây mô hình này đang được đánh giá lại và một mô hình mới được xây dựng có tính mở rộng và bảo trì được xây dựng: mô hình n-lớp. Trong đó, giải pháp mô hình 4-lớp được đưa ra bằng cách sử dụng một lớp trung gian giữa lớp logic và lớp lưu trữ. Một máy chủ ứng dụng trong mô hình n-lớp sẽ cung cấp các API phục vụ các chức năng, hoạt động và được sử dụng bởi ứng dụng. Các máy chủ có thể sử dụng nhiều loại ngôn ngữ. nhiều loại hệ quản trị CSDL khác nhau.



Hình 2: Mô hình ứng dụng 4-lớp.

Mỗi lớp có thể được đặt trên nhiều máy chủ hoặc cùng một máy chủ. Cũng như một máy chủ có thể đặt nhiều lớp. Việc chia thành nhiều lớp nhỏ hơn cho phép dễ dàng mở rộng, phân tách rõ hơn giữa các LTV, làm cho mã nguồn ứng dụng dễ đọc hơn, cũng như dễ dàng tái sử dụng. Phương pháp này cũng loại bỏ các điểm yếu duy nhất của ứng dụng. Hiện nay, mô hình 3-lớp và 4-lớp đang rất phổ biến trên Internet. Tuy nhiên, như đã nói mô hình n-lớp rất linh hoạt và sẽ cung cấp vô số cách để chia tách và triển khai.

### Lỗ hổng SQL injection.

Các ứng dụng web ngày càng trở nên phức tạp hơn. Phạm vi cũng mở rộng từ Internet đến nội bộ, từ các hệ thống thương mại điện tử đến các hệ thống đối tác, hệ thống quản lý. Sự đa dạng của các hệ thống, sự nhạy cảm của các dữ liệu trở nên quan trọng với các doanh nghiệp. Các ứng dụng web cũng vô cùng đa dạng và biến đổi trong các dòng mã nguồn, càng nhiều tính năng càng làm tăng khả năng mắc lỗi của ứng dụng. Ngoài ra, sự phát triển của ngành An toàn thông tin (ATTT), hacker ngày càng tập trung vào việc khai thác, chiếm quyền điều khiển các hệ thống.

Tấn công SQL injection là một kiểu tấn công mà câu lệnh truy vấn SQL được tạo thành từ đầu vào phía người dùng, sau đó chuyển tới máy chủ SQL phía back-end để phân tích và thực thi. Bất kỳ thủ tục để xây dựng mệnh đề truy vấn SQL cũng có khả năng mắc lỗi, giống như sự đa dạng trong các phương pháp lập trình. Một hình thức cơ bản của SQL injection là chèn trực tiếp đoạn mã vào tham số (parameter) được ghép nối với câu truy vấn SQL và thực thi. Một dạng khác ít trực tiếp mà mã tấn công sẽ được tiêm (inject) vào trong một bảng, khi chuỗi này được lấy ra và ghép vào một câu truy vấn SQL động, nó sẽ được thực thi. Khi mà ứng dụng web không thực hiện tiền xử lý – sanitize đầu vào, thành phần sẽ được đưa vào câu truy vấn SQL động, kẻ tấn công sẽ tạo được câu lệnh SQL mong muốn.

Để minh họa, chúng ta quay lại với ví dụ cửa hàng trực tuyến, cho phép hiển thị các sản phẩm có giá nhỏ hơn 100$:

<http://www.victim.com/products.php?val=100>.

Lần này, chúng ta sẽ thêm câu lệnh SQL bằng cách thêm vào tham số val đoạn mã: ‘OR ‘1’=‘1. URL sẽ thành:

http://www.victim.com/products.php?val=100’ OR ‘1’=‘1

Mã PHP sẽ thực thi và trả về tất cả các sản phầm, không liên quan đến giá của nó. Bởi vì chúng ta đã thay đổi logic của câu truy vấn SQL, khi nối thêm toán hạng OR, điều đó làm cho câu truy vấn luôn trả về True (vì thật sự 1 luôn bằng 1). Câu lệnh sau sẽ được sinh ra thực thi:

|  |
| --- |
| SELECT ∗  FROM ProductsTbl  WHERE Price < ‘100.00’ OR ‘1’ = ‘1’  ORDER BY ProductDescription; |

Có rất nhiều cách để khai thác lỗ hổng SQL injection để đạt được vô số các mục tiêu. Sự thành công của các cuộc tấn công thường là phụ thuộc nhiều vào cơ sở dữ liệu và kết nối các hệ thống cơ bản mà đang bị tấn công. Đôi khi nó có thể mất rất nhiều kỹ năng và sự kiên trì để khai thác một lỗ hổng tiềm năng.

Trong ví dụ trên, chúng ta đã thấy được cách kẻ tấn công tạo ra câu lệnh câu lệnh truy vấn SQL động từ đầu vào không được kiểm tra, từ đó thực thi các hành vi phía LTV không lường trước được.Trong ví dụ sau, một CMS (Content Management System) là một ứng dụng web được sử dụng để tạo, chỉnh sửa, quản lý và xuất bản nội dung cho một trang web mà không cần có một sự hiểu biết sâu hoặc khả năng mã trong HTML. Sử dụng các URL sau đây để truy cập các ứng dụng CMS:

<http://www.victim.com/cms/login.php?username=foo&password=bar>

Ứng dụng CMS yêu cầu cung cấp một tên người dùng và mật khẩu hợp lệ trước khi có thể truy cập vào các chức năng của nó. Truy cập các URL trước sẽ cho kết quả trong các lỗi “Incorrect username or password, please try again.”. Dưới đây là các mã cho file login.php:

|  |
| --- |
| // connect to the database  $conn = mysql\_connect(“localhost”,“username”,“password”);  // dynamically build the sql statement with the input  $query = “SELECT userid FROM CMSUsers WHERE user = ‘$\_GET[“user”]’ ”.  “AND password = ‘$\_GET[“password”]’”;  // execute the query against the database  $result = mysql\_query($query);  // check to see how many rows were returned from the database  $rowcount = mysql\_num\_rows($result);  // if a row is returned then the credentials must be valid, so  // forward the user to the admin pages  if ($rowcount ! = 0){header(“Location: admin.php”);}  // if a row is not returned then the credentials must be invalid  else {die(‘Incorrect username or password, please try again.’)} |

File login.php tự động tạo ra một câu lệnh SQL sẽ trả về một bản ghi mà nếu một tên truy nhập và mật khẩu hợp lệ được nhập vào. Các truy vấn sẽ trả userid tương ứng cho người sử dụng nếu giá trị user và password phù hợp với một giá trị được lưu trữ tương ứng trong bảng CMSUsers:

|  |
| --- |
| SELECT userid  FROM CMSUsers  WHERE user = ‘foo’ AND password = ‘bar’; |

Vấn đề là trong đoạn mã trên, lập trình viên (LTV) tin rằng số lượng bản ghi trả về chỉ có thể là 0 hoặc 1. Trong ví dụ trước, chúng ta đã inject để kết quả trả về luôn là TRUE. Nếu tiếp tục sử dụng lại, chúng ta có thể làm sai logic của phần xác thực. Bằng cách thêm ‘OR ‘1’=’1 vào sau URL, đoạn script PHP trong lần này sẽ trả về tất cả user trong bảng CMSUsers. URL trông như sau:

http://www.victim.com/cms/login.php?username=foo&password=bar’ OR ‘1’=’1

Tất cả các userids được trả về bởi vì chúng ta đã thay đổi logic của câu truy vấn. Dưới đây là các truy vấn đã được xây dựng và thực hiện:

|  |
| --- |
| SELECT userid  FROM CMSUsers  WHERE user = ‘foo’ AND password = ‘password’ OR ‘1’ = ‘1’; |

Logic của ứng dụng có nghĩa là nếu CSDL trả về nhiều hơn 1 bản ghi, chúng ta đã nhập thông tin đăng nhập chính xác và có quyền truy cập vào file admin,php được bảo vệ. Sử dụng kỹ thuật khai thác này, thông thường sẽ được đăng nhập như là user đầu tiên trong bảng CMSUsers. Bằng việc khai thác lỗ hổng SQL injection đã phá vỡ cơ chế xác thực của ứng dụng.

### Nguyên nhân của lỗ hổng SQL injection.

SQL là ngôn ngữ chuẩn để truy cập vào các hệ quản trị CSDL: Microsoft SQL Server, Oracle, MySQL, Sybase, và Informix… Hầu hết các ứng dụng Web cần phải tương tác với một CSDL, và hầu hết các ngôn ngữ lập trình ứng dụng web, như ASP, C #, .NET, Java và PHP, cung cấp cách chương trình kết nối với một cơ sở dữ liệu và nó. Lỗ hổng SQL injection xảy ra khi các LTV ứng dụng Web không đảm bảo các giá trị nhận được từ người dùng (Web form, cookie, input parameter…) được kiểm tra chặt chẽ trước khi truyền vào câu truy vấn SQL và thực hiện trên máy chủ CSDL. Nếu một kẻ tấn công có thể kiểm soát đầu vào đó và thay đổi, những kẻ tấn công có thể thực thi câu lệnh SQL trên máy chủ CSDL.

Mỗi ngôn ngữ lập trình cung cấp một số cách khác nhau để xây dựng và thực thi câu lệnh SQL, và các LTV thường sử dụng kết hợp của các phương pháp khác nhau. Có rất nhiều trang web cung cấp các hướng dẫn và các ví dụ để giúp các LTV giải quyết vấn đề thông thường, và thường các ví dụ này không an toàn. Nếu không có một sự hiểu biết về các hệ quản trị CSDL, nhận thức về các vấn đề bảo mật, LTV có thể làm ra một ứng dụng mắc lỗi SQL injection.

Dynamic string building là một kỹ thuật cho phép LTV tạo ra câu lệnh SQL trong khi chạy. Một câu lệnh SQL động được xây dựng ở thời gian thực hiện, đối với những điều kiện khác nhau tạo ra các câu lệnh SQL khác nhau. Nó có thể cho các LTV xây dựng ra các câu lệnh tự động, tùy thuộc vào hoàn cảnh, đầu vào từ người dùng. Các nguyên nhân dẫn đến việc tạo ra câu lệnh SQL không an toàn có thể là:

* Không kiểm soát ký tự Escape.
* Không kiểm soát loại dữ liệu.
* Không kiểm soát quá trình sinh câu lệnh SQL.
* Không kiếm soát thông báo lỗi.

Nếu không kiểm soát chặt quá trình sinh câu lệnh này, kẻ tấn công hoàn toàn có thể tạo ra câu lệnh SQL và thực thi trên máy chủ CSDL.

### Ảnh hưởng của lỗ hổng SQL injection với tổ chức.

Khai thác lỗ hổng SQL injection, kẻ tấn công có thể chiếm quyền điều khiển toàn bộ CSDL. Kẻ tấn công có thể lấy toàn bộ các thông tin nhạy cảm: thông tin tài khoản, thông tin thẻ tín dụng. Một số có thể chỉnh sửa, tạo thêm các tài khoản quản trị mức ứng dụng để từ đó chiếm quyền điều khiển ứng dụng, thực thi trái phép các hành vi không được phép.

Một hướng khác, kẻ tấn công thực hiện tấn công nâng quyền. Một số hệ quản trị CSDL cho phép thực thi câu lệnh hệ điều hành. Thông thường, các hệ quản trị CSDL này được chạy với quyền quản trị, do đó, có thể thực thi các câu lệnh hệ điều hành với quyền quản trị. Khai thác theo hướng này, kẻ tấn công hoàn toàn có thể chiếm quyền điều khiển hoàn toàn máy chủ.

Một số nguồn trên thế giới có thể cho thấy sự ảnh hưởng của lỗ hổng SQL injection. Năm 2011, CWE (Common Weakness Enumeration)/SANS Top 25 Most Dangerous Software Errors là một danh sách thống kê các lỗ hổng nghiêm trọng nhất của phần mềm. Danh sách được tổng hợp từ hơn 20 nguồn khác nhau, sử dụng cơ chế Common Weakness Scoring System (CWSS) để chấm điểm. Vào năm 2011, lỗ hổng SQL injection đứng đầu danh sách (<http://cwe.mitre.org/top25/index.html>).

Ngoài ra, Open Web Application Security Project (OWASP) cũng liệt kê lỗi nhúng mã (bao gồm SQL injection) là lỗ hổng bảo mật nghiêm trọng nhất trong top 10 lỗ hổng ứng dụng web năm 2010. Mục đích chính của OWASP Top 10 là để đào tạo LTV, quản lý, thiết kế phần mềm, và các tổ chức về những hậu quả của các lỗ hổng bảo mật ứng dụng web phổ biến nhất. Trong danh sách trước đó được phát hành năm 2007, lỗ hổng SQL injection đứng thứ hai. Tuy nhiên, vào năm 2010 khi thay đổi cách tính điểm dựa trên ảnh hưởng/nguy cơ rủi ro, lỗ hổng SQL injection đã lên đầu.

Một nguồn khác, đó là thông tin các website bị tấn công được thống kê trên Zone-H. Một lượng lớn các trang web bị tấn công khai thác bằng lỗ hổng SQL injection. Giới truyền thông cũng có nhiều công bố về các vụ tấn công ảnh hưởng đến các công ty, tổ chức lớn:

Tháng 2/2002, Jeremiah Jacks ([www.securityfocus.com/news/346](http://www.securityfocus.com/news/346)) phát hiện lỗ hổng SQL injection của Guess.com, thu được ít nhất thông tin thẻ tín dụng của 200,000 khách hàng.

Tháng 6/2003, Jeremiah Jacks tiếp tục tấn công PetCo.com ([www.securityfocus.com/news/6194](http://www.securityfocus.com/news/6194)), thu được thông tin thẻ tín dụng của 500.000 khách hàng qua lỗ hổng SQL injection.

Ngày 17 tháng 6 năm 2005, Master Card cảnh báo khách hàng có sự vi phạm bảo mật trong hệ thống Card Systems Solutions, đây cũng là vụ tấn công lớn nhất được ghi nhận liên quan đến SQL injection ([www.ftc.gov/os/caselist/0523148/0523148complaint.pdf](http://www.ftc.gov/os/caselist/0523148/0523148complaint.pdf)), kẻ tấn công đã tiếp cận hơn 40 triệu thẻ tín dụng.

Tháng 8 năm 2007, United Nations Web site (www.un.org) bị thay đổi giao diện thông qua lỗ hổng SQL injection, hiển thị thông điệp bài Mỹ (<http://news.cnet.com/8301-10784_3-9758843-7.html>).

Tháng 2 năm 2009, một nhóm hacker Rumai sử dụng lỗ hổng SQL injection khai thác Web sites Kaspersky, F-Secure, và Bit-Defender. Nhóm hacker này cũng đã từng tấn công các website lớn như: RBS WorldPay, CNET.com, BT.com, Tiscali.co.uk, và national-lottery.co.uk.

Tháng 4 năm 2011, Barracuda Networks Web site (barracudanetworks.com) mắc lỗ hổng SQL injection, toàn bộ CSDL bị công khai gồm cả thông tin đăng nhập và mật khẩu đã mã hóa.

Tháng 5 năm 2011, LuzSec chiếm quyền một loạt website của Sony (sonypictures.com, SonyMusic.gr, and SonyMusic.co.jp). LulzSec tuyên bố có thể mật khẩu, email hàng triệu khách hàng của Sony

Nếu như trước đây, các hacker thường chiếm quyền điều khiển Website để ghi điểm với các nhóm hacker khác, hoặc truyền bá, để lại một tin nhắn cụ thể. Tuy nhiên, ngày nay các hacker khai thác với mục đích cụ thể, thường là tài chính. Việc tấn công ngày càng được thực hiện âm thầm, lặng lẽ. Đồng nghĩa với đó là hậu quả của tấn công ngày càng nghiêm trọng.

## Tấn công Blind SQL injection.

# Chương 2: Các phương pháp tối ưu hóa tấn công Blind SQL injection

## Hạn chế của tấn công Blind SQL injection