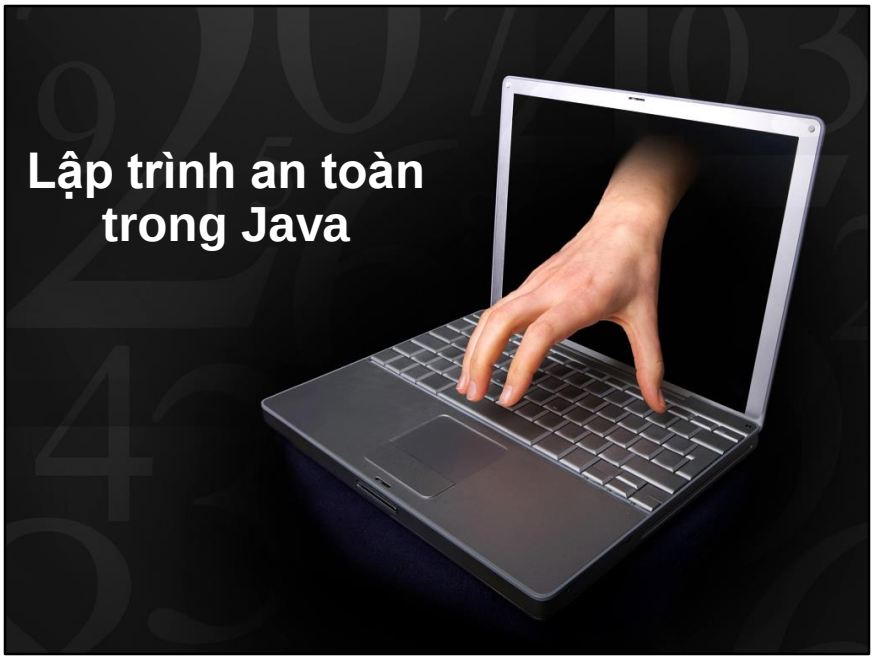


Nội dung

- Injection
- Inclusion

cuu duong than cong . com

4. Injection và Inclusion

Lỗi định dạng văn bản → thay đổi quyền kiểm soát khi thông dịch

- 1 dạng tấn công phổ biến
- Do chương trình thông dịch dữ liệu độc hại, thường là các định dạng văn bản.
- Gây ra thay đổi quyền điều khiển không mong đợi

4.1 – Tạo định dạng hợp lệ

- Sử dụng lỗi input để tạo ra định dạng không chính xác của output [1]:
  - Ký tự đặc biệt
  - Ký tự thoát không chính xác
  - Xóa bỏ một phần ký tự đặc biệt

Incorrect escaping: thêm dấu \ như " → \", cần escape lại không escape.

4.1 – Tạo định dạng hợp lệ

Ký tự thoát trong XML:

Original character	Escaped character
“	&quot;
‘	&apos;
<	&lt;
>	&gt;
&	&amp;

quote
a'postrophe
‘ampersand

4.1 – Tạo định dạng hợp lệ

- **Giải pháp:**
  - Phân tích và chuẩn hóa trước khi xác thực
  - Xóa bỏ dữ liệu không hợp lệ
  - Sử dụng thư viện đã được kiểm tra

- Nếu chuỗi đầu vào có định dạng đặc biệt, kết hợp hiệu chỉnh và xác thực → khả năng mắc lỗi cao -> Phân tích và chuẩn hóa nên được thực hiện trước khi xác thực.
- Nếu có thể hãy loại bỏ dữ liệu không hợp lệ và bất kỳ dữ liệu theo sau, mà không cố gắng sửa chữa, hiệu chỉnh.

Ví dụ, nhiều giao thức mạng dễ bị tấn công **cross-site POST**, bằng cách thông dịch phần HTTP body mặc dù HTTP header chứa lỗi.

- Sử dụng thư viện đã được kiểm tra thay vì viết lại.

Tạo XML từ văn bản thô (raw) dễ bị lỗi. Có rất nhiều thư viện để tạo XML. Tuy nhiên, đối với các định dạng khác thường không có thư viện thích hợp, như file cấu hình. → Tạo ra class xử lý một cách rõ ràng tất cả các định dạng của mã nguồn.

4.2 – Tránh SQL động

Lệnh SQL tạo ra từ input không tin cậy → injection

Giải pháp:

- Tránh SQL động
- Sử dụng Java Database Connectivity (JDBC) để tham số hóa biểu thức SQL
  - ✓ java.sql.PreparedStatement
  - ✓ java.sql.CallableStatement
  - ✓ ~~java.sql.Statement~~

SQL Injection

Dạng này thường xuất hiện khi Input chứa dấu nháy đơn (').

Sử dụng thư viện được viết sẵn, cấp cao hơn để cách ly mã nguồn ứng dụng với SQL.

Khi sử dụng thư viện không nhất thiết hạn chế các ký tự như dấu nháy (').

Nếu văn bản dùng cho XML / HTML được xử lý một cách chính xác trong output thì không cần thiết cấm các ký tự như (<) ở đầu vào SQL.

4.2 – Tránh SQL động

Mã nguồn mẫu:

```
String sql = "SELECT * FROM User WHERE userId = ?";
PreparedStatement stmt = con.prepareStatement(sql);
stmt.setString(1, userId);
ResultSet rs = stmt.executeQuery();
```


cuu duong than cong . com

JDBC

- Là chuẩn kết nối CSDL, cung cấp các interface & class để Java tương tác với CSDL
- JDBC driver: tập các class thực thi
- Có 4 loại JDBC driver:
  - JDBC-ODBC Bridge Driver
  - JDBC-Native API
  - Open Protocol-Net
  - 100% Pure Java (Proprietary-Protocol Net)

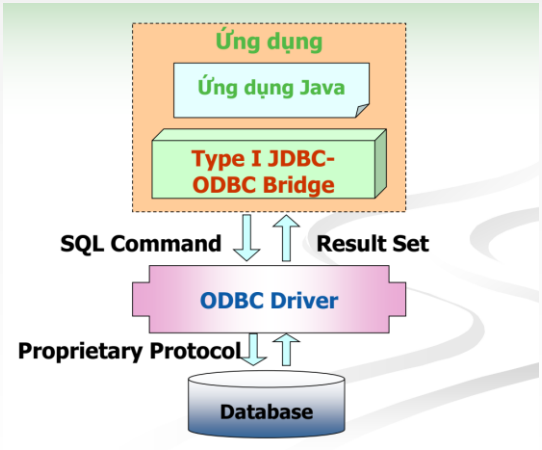
JDBC là chuẩn kết nối CSDL, cung cấp các interface & class nhằm tạo cơ sở cho các ứng dụng Java tương tác với các hệ quản trị CSDL
Tập hợp các lớp thực thi theo chuẩn JDBC để tương tác với 1 CSDL cụ thể gọi là JDBC driver.
Phần lớn ý tưởng của JDBC kế thừa từ chuẩn kết nối ODBC (Open Database Connectivity) của Microsoft.

Loại 1: JDBC-ODBC Bridge Driver

- Được cung cấp miễn phí bởi Sun-jdk
- Có thể truy xuất bất kỳ DBMS nào được hỗ trợ bởi ODBC driver
- Tính khả chuyển cao nhưng kém hiệu quả

cuu duong than cong . com

Loại 1: JDBC-ODBC Bridge Driver



cuu duong than cong . com

Loại 1: JDBC-ODBC Bridge Driver

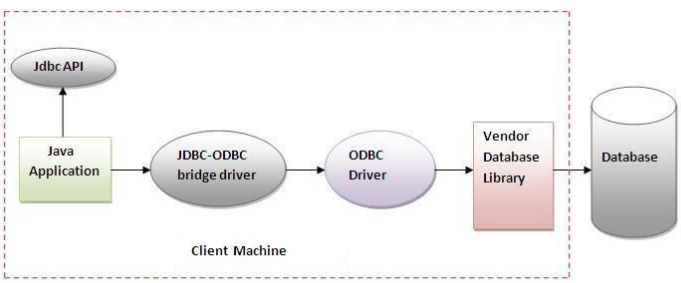


Figure- JDBC-ODBC Bridge Driver

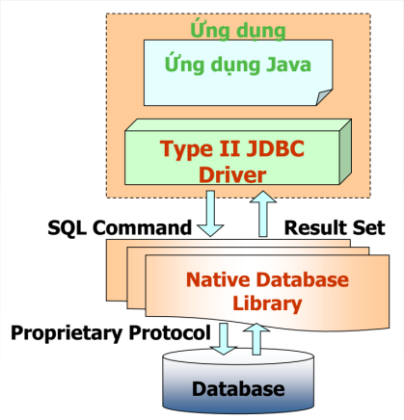
An toàn thông tin - UIT 12

- Sử dụng ODBC driver để connect DB.
- JDBC-ODBC bridge driver convert phương thức JDBC → phương thức OBDC.

Ưu điểm: Dễ sử dụng, dễ connect đến bất kỳ DB.
Nhược điểm: Hiệu suất kém do convert, cần phải cài đặt ODBC driver trên client-side.

Loại 2: JDBC-Native API

JDBC driver tương tác trực tiếp với database API.



An toàn thông tin - UIT 13

- 1 phần mã Java.
- 1 phần mã tự nhiên của DBMS (C/C++).

Loại 2: JDBC-Native API

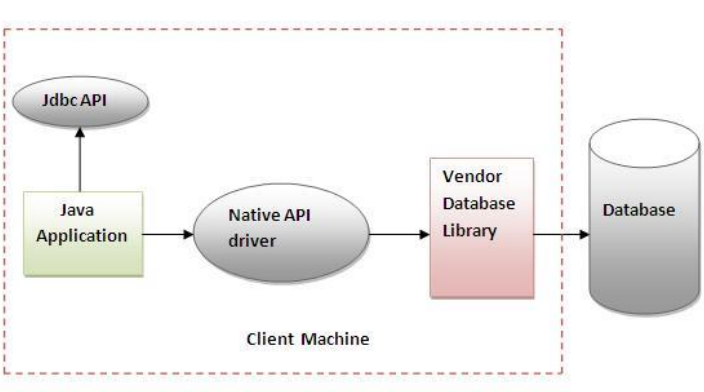
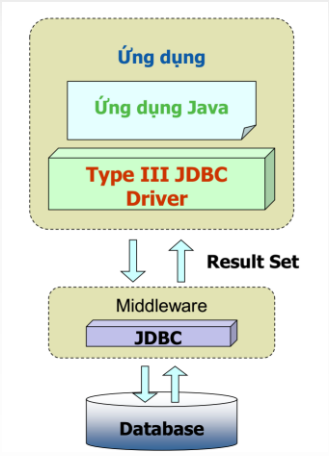


Figure- Native API Driver

- Native API driver sử dụng thư viện client-side của DB.
- Convert phương thức JDBC thành lời gọi native của API DB (C/C++).

Ưu điểm: hiệu suất hơn loại 1.
Khuyết điểm: cần cài cả thư viện Native driver và Vendor client trên client-side.

Loại 3: Open Protocol-Net



Tương tác với nhiều DBMS theo giao thức mở.

- 100% Java code.
- Cài đặt driver cả 2 phía client & server.

Loại 3: Open Protocol-Net

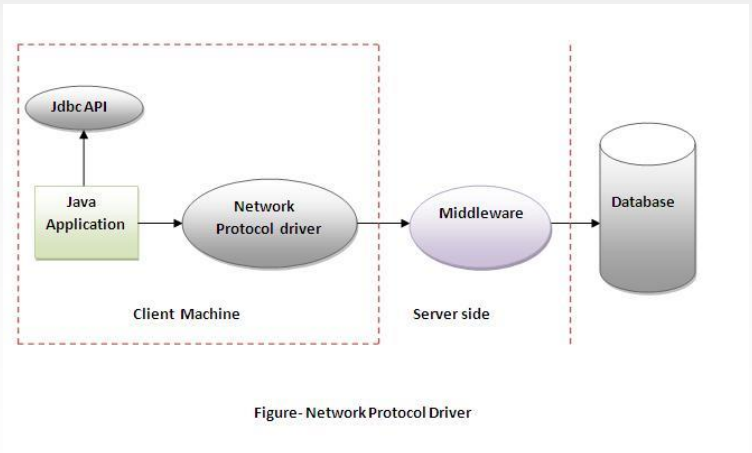


Figure- Network Protocol Driver

Network Protocol driver sử dụng middleware (application server) converts lời gọi JDBC trực tiếp hoặc gián tiếp thành vendor-specific database protocol.
Được viết hoàn toàn bằng ngôn ngữ Java.

Ưu điểm: không cần cài đặt thư viện ở client vì ứng dụng server có thể thực hiện many tasks như auditing, load balancing, logging,...
Nhược điểm:

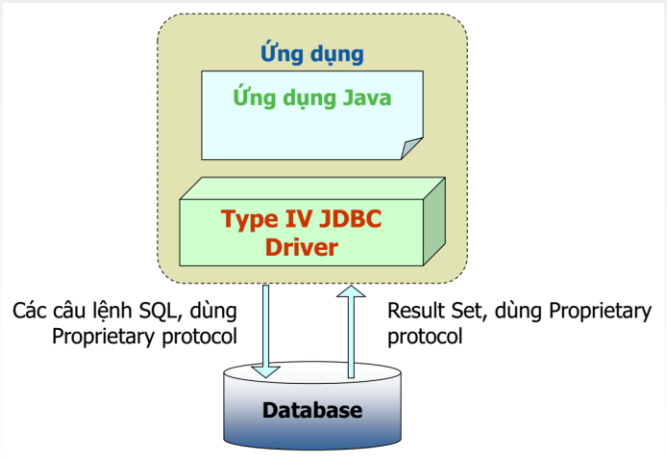
- Yêu cầu mạng trên máy client.
- Yêu cầu mã nguồn database riêng để được thực hiện trong lớp trung gian.
- Bảo trì Network Protocol driver tốn nhiều chi phí vì yêu cầu mã nguồn database-specific để được thực hiện trong lớp trung gian.

Loại 4: Proprietary-Protocol Net

- 100% java
- Truy xuất trực tiếp DBMS theo giao thức độc quyền
- Hiệu quả nhất

cuu duong than cong . com

Loại 4: Proprietary-Protocol Net



cuu duong than cong . com

Loại 4: Proprietary-Protocol Net

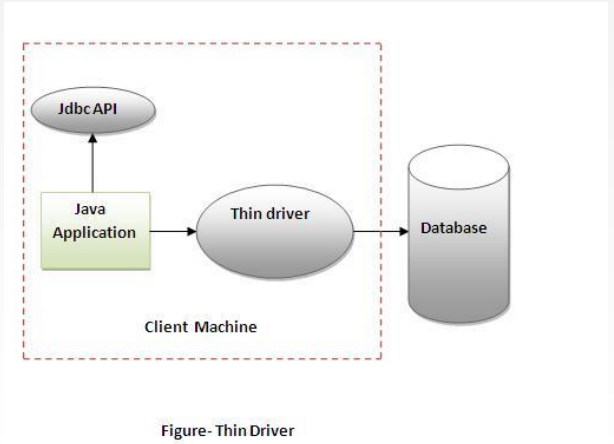


Figure- Thin Driver

Converts JDBC calls trực tiếp thành giao thức vendor-specific database, được viết hoàn toàn bằng Java.

Ưu điểm:

- Hiệu suất tốt nhất.
- Không cần phần mềm trên cả client và server.

Nhược điểm:

Driver phụ thuộc Database.

7 bước kết nối với JDBC

- Nạp driver
- Định nghĩa Connection URL
- Kết nối CSDL bằng đối tượng Connection
- Tạo đối tượng Statement
- Thực thi câu truy vấn
- Xử lý kết quả
- Đóng kết nối

cuu duong than cong . com

Nên sử dụng loại driver nào?

- Nếu truy cập 1 loại CSDL: Oracle, Sybase hay IBM → 4
- Nếu truy cập nhiều loại CSDL cùng lúc → 3
- Loại 2: khi loại 3 và 4 không hỗ trợ
- Loại 1: không dùng khi triển khai, thường cho test

cuu duong than cong . com

4.3 – Tạo XML và HTML cần phải thận trọng

- Dữ liệu không tin cậy nên được làm sạch trước khi đưa vào output XML/HTML.
- Vấn đề bảo mật:
  - XSS
  - Lỗ hỏng XML Injection

Lưu ý:

Đặc biệt cẩn thận khi sử dụng JSP.

- Dữ liệu không tin cậy nên được làm sạch một cách chính xác trước khi được đưa vào trong output HTML hay XML.
- Nếu không thực sự làm sạch các dữ liệu có thể dẫn đến nhiều vấn đề bảo mật khác nhau như Cross-Site Scripting (XSS) và các lỗ hổng Injection XML.
- Điều quan trọng là phải đặc biệt cẩn thận khi sử dụng Java Server Pages (JSP) – dùng xây dựng web app, hỗ trợ JDBC API cho thương mại.

4.3 – Tạo XML và HTML cần phải thận trọng

Giải pháp:

- Lọc – Filter
- Thoát – Escape
- Mã hóa – Encode

Làm cách nào để đạt hiệu quả cao?

- Liệt kê tất cả ký tự có tiềm năng để xử lý
- Cho phép ký tự an toàn, còn lại xử lý tất cả

Lưu ý:

Nên sử dụng thư viện để thực hiện.

Có nhiều cách để làm sạch dữ liệu trước khi đưa nó vào output.

- Những ký tự không chắc cho type cụ thể của output có thể được lọc, escaped hoặc mã hóa.
- Các ký tự được biết là an toàn có thể được cho phép và mọi thứ khác có thể được lọc, escaped, hoặc mã hóa.

+ Cách 1: tất cả ký tự đặc biệt (yêu cầu liệt kê tất cả ký tự có tiềm năng gây hại).
+ Cách 2: cho phép ký tự an toàn, còn lại sanitize tất cả (được ưa chuộng hơn) → thích hợp hơn, vì nó không đòi hỏi việc xác định và liệt kê tất cả các ký tự có tiềm năng gây ra vấn đề (dễ bỏ sót).

Làm sạch và mã hóa dữ liệu là việc phức tạp và dễ lỗi (không nên tin bản thân, dễ sót) → sử dụng thư viện để thực hiện trong lúc xây dựng HTML và XML.

4.4 – Tránh mọi dữ liệu không tin cậy trên command line

Dữ liệu độc hại có thể chuyển một tham số đơn như 1 option hoặc 2 tham số riêng biệt.

Giải pháp:

- Tham số được mã hóa
- Trong một file tạm
- Qua kênh an toàn

Khi tạo ra các tiến trình mới, không đặt bất kỳ dữ liệu không tin cậy trên cửa sổ dòng lệnh.
Dữ liệu độc hại có thể thông dịch 1 tham số đơn như 1 option hoặc như 2 tham số riêng biệt.

Bất kỳ dữ liệu cần được truyền đến các tiến trình mới nên được truyền dưới dạng:
+ Tham số được mã hóa (Base64).
+ Trong một tập tin tạm thời.
+ Thông qua một kênh được kế thừa.

4.5 – Hạn chế XML inclusion

XML Document Type Definitions (DTDs) cho phép URL định nghĩa như thực thể hệ thống:

- Tập tin cục bộ
- HTTP URL

- Tấn công:
  - XML External Entity (XXE)
  - Sử dụng XInclude

DTD cho phép các URL được định nghĩa như các thực thể hệ thống:
- Các tập tin cục bộ.
- HTTP URL trong mạng nội bộ hay localhost.

Tấn công:
- XML External Entity (XXE) chèn tập tin local vào dữ liệu XML có thể truy cập đến client khác.
- Tấn công tương tự có thể được thực hiện bằng cách sử dụng Xinclude: XSLT document(), XSLT import và XSLT include.

Mục đích của DTD là để định nghĩa cấu trúc của XML.

4.5 – Hạn chế XML inclusion

<xsl:value-of
select="**document**('celsius.xml')/celsius/result[@
value=$value]"/>

<xsl:**import** href="URI"/>

<xsl:**include** href="URI"/>

cuu duong than cong . com

4.5 – Hạn chế XML inclusion

- **Giải pháp:**

  Hạn chế đặc quyền và hạn chế tối đa khả năng cấu hình

- Cách an toàn nhất để tránh những vấn đề này trong khi duy trì hiệu năng của XML và để sử dụng khả năng hạn chế cấu hình tối đa cho XML parser là giảm đặc quyền.
- Hạn chế đặc quyền vẫn cho phép cấp quyền 1 số truy cập: việc inclusion đến các trang web cùng nguồn gốc.
- XML parser cũng có thể được cấu hình để hạn chế chức năng dựa trên những thứ được yêu cầu như không cấp phép cho những thực thể bên ngoài.

Lưu ý rằng: vấn đề này thường áp dụng cho việc sử dụng các API sử dụng XML nhưng không cụ thể API XML.

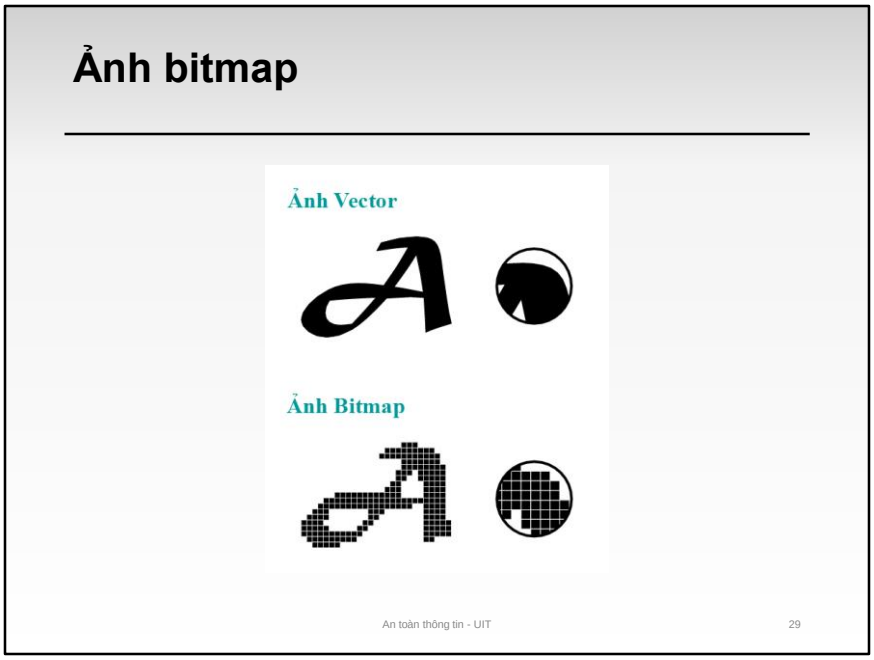
4.6 – Cẩn thận với những tập tin BMP

Ảnh BMP thường chứa tham chiếu đến tập tin ICC cục bộ → có thể đọc được nội dung.

Giải pháp:

- Tránh dùng ảnh BMP
- Giảm đặc quyền

International Color Consortium (File lưu chuẩn màu).
Trong khi nội dung của file ICC dường như không được quan tâm, hành động cố gắng để đọc các tập tin có thể là một vấn đề.



Bitmap: là ảnh được tạo bởi ma trận các điểm ảnh, tức pixel, để biểu thị hình ảnh. Mỗi pixel (điểm ảnh) được gán một vị trí và gán giá trị màu cụ thể. Thực tế, khi hiệu chỉnh, ta hiệu chỉnh điểm ảnh chứ không phải hiệu chỉnh đối tượng hay hình dạng.

Ảnh Vector được tạo bởi các đoạn thẳng và đường cong được định nghĩa bằng các đối tượng toán học gọi là Vector.
Hình Vector mô tả hình ảnh dựa trên các thuộc tính hình học của hình ảnh đó.
➔ **Chỉnh ảnh Vector là chỉnh thuộc tính của đối tượng đó.**

4.7 – Vô hiệu hóa hiển thị HTML trong Swing

Swing pluggable look-and-feels thông dịch **text** trong thành phần bắt đầu <html> như HTML.

→ thực hiện tấn công inclusion

→ Vô hiệu hóa thuộc tính HTML render

```
label.putClientProperty("html.disable", true);
```


Pluggable Look-and-Feel là một trong số những tính năng của Swing → cho phép ứng dụng Swing có thể thay đổi toàn bộ giao diện chỉ với một hai dòng code.
"Look" = GUI của component, "Feel" = behave (ví dụ như hiệu ứng khi hover, khi click,…).

Nếu text là từ một nguồn không tin cậy, kẻ xấu có thể xây dựng được mã HTML để các thành phần khác hiện thị hoặc thực hiện các cuộc tấn công inclusion.

Để disable đặc trưng render HTML, set thuộc tính "html.disable" của mỗi component thành Boolean.TRUE.

4.8 – Cẩn thận khi thông dịch mã nguồn không tin cậy

- Nếu mã nguồn không tin cậy hỗ trợ mã tin cậy thì nên chạy trong trong sanbox
- Thành phần và API có thể thực thi mã không tin cậy:
  - Script chạy bằng javax.script (API)
  - JavaScript chạy trên trình duyệt (thường không được đăng ký xác thực)
  - Thực thi của thông dịch XSLT (Oracle)
  - Java Sound

  …

Mã nguồn có thể được ẩn giấu nhiều nơi. Nếu mã nguồn không tin cậy được dùng để hỗ trợ mã nguồn gốc thì sandbox phải được xây dựng để chạy nó trong đó.

Các thành phần, API có tiềm năng có thể thực thi mã không tin cậy bao gồm:

- Script chạy qua các API javax.script hoặc tương tự.
- Giao diện LiveConnect với JavaScript chạy trong trình duyệt. JavaScript chạy trên một trang web sẽ thường không được xác thực với chứng nhận ký mã đối tượng.
- Theo mặc định, thực thi Oracle của trình phiên dịch XSLT cho phép extensions gọi mã Java. Thiết lập tính năng javax.xml.XMLConstants.FEATURE_SECURE_PROCESSING để vô hiệu hóa nó.
- Java Sound sẽ nạp mã qua các phương thức javax.sound.midi.MidiSystem.getSoundbank.
- RMI có thể cho phép tải các đoạn mã từ xa được thiết lập bởi kết nối từ xa. Trên Oracle JDK, điều này mặc định được vô hiệu thóa nhưng có thể được kích hoạt hay vô hiệu hóa thông qua thuộc tính hệ thống java.rmi.server.useCodebaseOnly.

- LDAP (RFC 2713) cho phép nạp mã từ xa trong phản hồi máy chủ. Trên Oracle JDK, điều này mặc định được tắt nhưng có thể được kích hoạt hay vô hiệu hóa thông qua thuộc tính hệ thống com.sun.jndi.ldap.object.trustURLCodebase.
- Nhiều thực thi SQL cho phép thực thi mã với các ảnh hưởng bên ngoài của cơ sở dữ liệu riêng của mình.

Sandbox là một kỹ thuật giúp cô lập các ứng dụng, ngăn chặn các phần mềm độc hại làm hỏng hệ thống máy tính hay cài mã độc đánh cắp thông tin cá nhân của bạn.

Ví dụ: yêu cầu dùng camera của trình duyệt, dùng tài nguyên khi cài app điện thoại,…

cuu duong than cong . com

4.9 – Ngăn tấn công qua chèn số kiểu float

- NaN hay giá trị không xác định được đưa vào từ input không tin cậy thường không được chú ý để làm sạch
- -∞ và +∞ không thể biểu diễn trong khu vực bộ nhớ
- NaN có thể được tạo ra
  - 0.0 / 0.0
  - ∞ – ∞

Làm việc với số chấm động đòi hỏi phải cẩn thận khi nhập những số đó từ bên ngoài vào ranh giới tin cậy.

- NaN (Not A Number, không phải là số) hay giá trị vô hạn có thể được inject vào các ứng dụng thông qua dữ liệu đầu vào không tin cậy
- Chuyển đổi **Strings** (không tin cậy) **qua phương thức Double.valueOf**.

Thường giá trị ngoại lệ không được chú ý để xử lý làm sạch trước khi thực thi.

4.9 – Ngăn tấn công qua chèn số kiểu float

- Cẩn thận khi chuyển số chấm động ngoại lệ về dạng số nguyên thủy
  - Chuyển NaN về integer: 0
  - Vô cùng dương về integer: Integer.MAX_VALUE.
  - Vô cùng âm về integer: Integer.MIN_VALUE
- Đối với những ứng dụng yêu cầu giá trị chấm động → lưu trong thành phần cục bộ và lọc giá trị trước khi truyền lại cho ứng dụng.

Cần cẩn thận khi chuyển số chấm động ngoại lệ về dạng hỗ trợ sẵn như short, integer và long.
Chuyển đổi → kết quả có thể không chính xác trong trường hợp sử dụng cụ thể.

Có những ứng dụng riêng yêu cầu các giá trị ngoại lệ, như phân tích **số liệu khoa học** dựa vào quá trình xử lý số học. Tuy nhiên, các giá trị kết quả cho mục đích này nên được chứa trong thành phần cục bộ. Điều này có thể có được bằng cách làm sạch bất kỳ kết quả chấm động trước khi chuyển chúng trở lại những phần chung của một ứng dụng.

4.9 – Ngăn tấn công qua chèn số kiểu float

- Class Double và Float cung cấp các hàm lọc:
  - isNan
  - isInfinte

Lưu ý:

Double.NaN == value luôn trả về False

Class Double và Float có cung cấp các phương thức isNaN và isInfinite để santinize.

Lưu ý:

So sánh các thể hiện của Double.NaN qua toán = luôn luôn kết quả là false → ảnh hưởng đến tìm kiếm trong **maps** hoặc **collections** khi sử dụng toán tử = được wrap trong phương thức so sánh trong định nghĩa class.

4.9 – Ngăn tấn công qua chèn số kiểu float

```
if (Double.isNaN(untrusted_double_value)) {
    // specific action for non-number case
}
if (Double.isInfinite(untrusted_double_value)) {
    // specific action for infinite case
}
// normal processing starts here
```


cuu duong than cong . com

Tóm tắt

- Định dạng dữ liệu chính xác trước khi sử dụng (XML)
- Tránh SQL động
- Làm sạch dữ liệu trước khi dùng để tạo XML hoặc HTML
- Tránh truyền dữ liệu từ nguồn không tin cậy trên command line
- Hạn chế XML inclusion
- Sử dụng ảnh Bitmap
- Chức năng hiển thị HTML trong package Swing
- Sử dụng mã từ nguồn không tin cậy để hỗ trợ
- Ngăn chặn việc tấn công qua chèn số kiểu float

cuu duong than cong . com