

STT	Người ký	Đơn vị	Thời gian ký	Ý kiến
1	Trần Mạnh Cường	Phó Tổng Giám đốc - VNPAY	28/11/2023 00:20:51	Tôi đồng ý !
2	Nguyễn Minh Tiến	Trưởng phòng - P.An ninh TT & QL tuân thủ	24/11/2023 09:46:39	-
3	Nguyễn Tiến Lịch	Phó phòng - P.An ninh TT & QL tuân thủ	16/11/2023 17:13:51	-

Hoàng Phương Linh linhhp@vnpay.vn 28/11/2023 08:29:08




# CÔNG TY CỔ PHẦN GIẢI PHÁP THANH TOÁN VIỆT NAM



## HƯỚNG DẪN TIÊU CHUẨN OBFUSCATION ỨNG DỤNG

Mã hiệu: HDCV-KCN-005

Phiên bản: 1.0


Người phê duyệt	Người xem xét	Người soạn thảo
		
<b>Phó Tổng Giám đốc Trần Mạnh Cường</b>	<b>Phó Giám đốc KCN Nguyễn Minh Tiến</b>	<b>Nguyễn Tiến Lịch</b>

39/HD-ANTT&QLTT-KCN ngày 28 tháng 11 năm 2023

Ký bởi: VP Trụ sở chính VNPAY

Ngày Ký: 28/11/2023; 07:47:31

Hà Nội, 12/2023

	<p style="text-align: center;"><b>HƯỚNG DẪN TIÊU CHUẨN OBFUSCATION ỨNG DỤNG</b></p>	<p>Mã hiệu: HD-KCN-005 Phiên bản: 1.0 Ngày hiệu lực: 01/12/2023</p>
<p style="text-align: center;"><b>Khối Công Nghệ</b></p>		

**THEO DÕI LỊCH SỬ SỬA ĐỔI**

STT	Nội dung	Ngày hiệu lực	Phiên bản
1	Ban hành mới	01/12/2023	v1.0

**MỨC BẢO MẬT**

**Lưu hành nội bộ:** Tài liệu này chứa các thông tin nội bộ của VNPAY. Nội dung của tài liệu này không được tiết lộ cho bất kỳ bên thứ ba nào nếu không có sự cho phép của VNPAY.

## MỤC LỤC

1 MỤC ĐÍCH VÀ PHẠM VI ÁP DỤNG	3
1.1 Mục đích	3
1.2 Phạm vi áp dụng	3
2 ĐỊNH NGHĨA VÀ TỪ VIẾT TẮT	3
3 NỘI DUNG	3
3.1 Định nghĩa Obfuscation	3
3.2 Vì sao phải obfuscation ứng dụng	3
3.3 Các tiêu chuẩn cho obfuscation	4
3.4 Các kỹ thuật Obfuscation	5

Hoàng Phương Linh linhhp@vnpay.vn 28/11/2023 08:29:08

## 1 MỤC ĐÍCH VÀ PHẠM VI ÁP DỤNG

### 1.1 Mục đích

Hướng dẫn này đưa ra các tiêu chuẩn obfuscation ứng dụng

### 1.2 Phạm vi áp dụng

Hướng dẫn này được áp dụng cho tất cả các phòng ban thuộc Khối Công Nghệ của công ty VNPAY.

## 2 ĐỊNH NGHĨA VÀ TỪ VIẾT TẮT

- **Công ty/ VNPAY:** Công ty Cổ phần Giải pháp Thanh toán Việt Nam – VNPAY
- **KCN:** Khối Công Nghệ
- **P. ANTT&QLTT:** Phòng An ninh thông tin và Quản lý tuân thủ
- **Obfuscation (obfuscate):** Thực hiện các biện pháp nhằm làm cho mã nguồn hoặc mã máy trở nên khó hiểu, khó đọc hoặc khó phân tích cho con người.
- **Reverse engineering:** Là quá trình phân tích một sản phẩm, hệ thống, hoặc công nghệ đã tồn tại để hiểu cách thức hoạt động và thiết kế của nó. Điều này thường được thực hiện để tìm ra cấu trúc, tính năng, và nguyên lý hoạt động của một sản phẩm hoặc hệ thống khi thông tin chi tiết về nó không được công bố hoặc không dễ dàng truy cập.

## 3 NỘI DUNG

### 3.1 Định nghĩa Obfuscation

- Obfuscation (hoặc obfuscate) là quá trình biến đổi mã nguồn một cách có chọn lọc để làm cho mã trở nên khó hiểu đối với người đọc hoặc người phân tích, trong khi vẫn duy trì chức năng của chương trình. Mục đích chính của obfuscation là bảo vệ mã nguồn khỏi việc reverse engineering, ngăn chặn việc sao chép không phép, hoặc che dấu thông tin nhạy cảm như khóa API, mật khẩu, hay logic quan trọng trong ứng dụng.
- Các kỹ thuật obfuscation có thể bao gồm việc thay đổi tên biến, hàm, lớp, mã hóa chuỗi, thêm mã không cần thiết, chuyển đổi cấu trúc điều khiển (control flow), hoặc mã hóa thông tin trong mã nguồn để làm cho chúng khó đọc và hiểu hơn.

### 3.2 Vì sao phải obfuscation ứng dụng

Việc sử dụng obfuscation trong phát triển phần mềm có một số lợi ích cụ thể như sau:

- **Bảo vệ mã nguồn:** Một trong những lý do chính để obfuscate ứng dụng là bảo vệ mã nguồn khỏi việc bị đọc hiểu hay reverse engineering. Khi mã nguồn trở nên khó đọc,

khó hiểu, việc tìm ra logic hoặc thuật toán cụ thể trở nên khó khăn hơn đối với những người không có kiến thức cụ thể hoặc công cụ phân tích.

- **Bảo vệ thông tin nhạy cảm:** Obfuscation có thể giúp bảo vệ thông tin nhạy cảm như khóa API, mật khẩu, hoặc thông tin định danh khác trong mã nguồn. Việc mã hóa hoặc che dấu thông tin này có thể ngăn chặn việc truy cập trực tiếp từ bên ngoài.
- **Phòng chống việc sao chép mã nguồn:** Khi mã nguồn trở nên khó hiểu, người dùng có thể gặp khó khăn trong việc sao chép hoặc sử dụng mã nguồn mà không có sự cho phép. Obfuscation có thể làm giảm nguy cơ này.
- **Bảo vệ quyền sở hữu trí tuệ:** Trong một số trường hợp, việc bảo vệ thuật toán hoặc logic đặc biệt có thể giúp bảo vệ quyền sở hữu trí tuệ, đặc biệt là khi có những tính năng độc quyền hoặc đặc điểm kỹ thuật độc đáo.
- **Phòng chống giao diện ngược (Reverse Engineering):** Obfuscation có thể làm giảm khả năng reverse engineering bằng cách làm cho mã nguồn hoặc mã máy trở nên khó đọc và hiểu hơn, từ đó ngăn chặn việc phân tích ứng dụng để tìm ra cách hoạt động nội bộ của nó.

### 3.3 Các tiêu chuẩn cho obfuscation

Khi thực hiện obfuscation, có một số tiêu chuẩn và hướng dẫn có thể được xem xét để thực hiện quy trình này một cách hiệu quả và an toàn. Dưới đây là một số tiêu chuẩn quan trọng cho việc obfuscation:

- **Bảo vệ thông tin nhạy cảm:** Đảm bảo rằng thông tin nhạy cảm như khóa API, mật khẩu, hay thông tin cá nhân được mã hóa hoặc che dấu một cách an toàn trong quá trình obfuscation.
- **Bảo vệ thuật toán quan trọng:** Các thuật toán quan trọng hoặc logic đặc biệt nên được bảo vệ thông qua obfuscation để ngăn chặn việc reverse engineering.
- **Không ảnh hưởng đến chức năng ứng dụng:** Obfuscation không nên ảnh hưởng đến chức năng hoạt động của ứng dụng. Đảm bảo rằng sau khi obfuscate, ứng dụng vẫn hoạt động đúng như dự kiến và không có lỗi xảy ra.
- **Bảo vệ quyền sở hữu trí tuệ:** Tuân thủ các quy định về bản quyền và quyền sở hữu trí tuệ. Đảm bảo rằng việc obfuscate không vi phạm các quy định hoặc quy tắc liên quan đến quyền sở hữu trí tuệ.

- **Tạo tài liệu chi tiết:** Tạo tài liệu chi tiết về quá trình obfuscation. Ghi chú về các thay đổi cụ thể đã được thực hiện và quá trình obfuscation nhằm hỗ trợ việc bảo trì và hiểu mã nguồn sau này.
- **Duy trì hiệu năng:** Obfuscation không nên ảnh hưởng đến hiệu suất của ứng dụng. Đảm bảo rằng việc obfuscate không làm giảm tốc độ hoạt động hoặc tăng thời gian phản hồi của ứng dụng.
- **Bảo mật cấp độ cao:** Sử dụng các phương pháp obfuscation có độ an toàn cao để đảm bảo rằng mã nguồn không dễ dàng bị giải mã hoặc phân tích.
- **Sự cân nhắc trong việc Obfuscate:** Xem xét kỹ lưỡng trước khi obfuscate và đánh giá lợi ích so với rủi ro có thể xảy ra. Tránh obfuscate quá nhiều có thể gây khó khăn trong việc bảo trì và phát triển sau này.
- **Xác định mục tiêu rõ ràng:** Đặt ra mục tiêu rõ ràng cho việc obfuscate: Muốn bảo vệ thông tin nhạy cảm, ngăn chặn reverse engineering, hay giữ riêng tư về logic quan trọng trong mã nguồn?
- **Obfuscate cẩn thận:** Lựa chọn cẩn thận các phần cần obfuscate. Hãy tập trung vào những phần quan trọng nhất của mã nguồn như: Thông tin nhạy cảm, thuật toán quan trọng, và logic chương trình.

### 3.4 Các kỹ thuật Obfuscation

Có nhiều kỹ thuật obfuscation khác nhau có thể được sử dụng để làm cho mã nguồn hoặc mã máy trở nên khó hiểu, khó đọc và khó phân tích. Dưới đây là một số kỹ thuật obfuscation phổ biến:

- **Mangling và Renaming (Thay đổi tên):** Thay đổi tên biến, hàm và lớp thành các tên ngẫu nhiên hoặc viết tắt mà vẫn giữ nguyên chức năng. Ví dụ, userPassword có thể được đổi thành a hoặc b.
- **String Encryption (Mã hóa chuỗi):** Mã hóa chuỗi trong mã nguồn để làm cho chúng khó đọc và không thể nhìn thấy trực tiếp từ mã nguồn. Khi cần sử dụng chuỗi đó, chúng sẽ được giải mã trong quá trình chạy.
- **Control Flow Obfuscation (Tối ưu hóa luồng điều khiển):** Thay đổi cấu trúc luồng điều khiển trong mã, ví dụ như thêm hoặc loại bỏ các lệnh nhảy (jump statements), chuyển đổi các cấu trúc điều khiển như if-else thành dạng phức tạp hơn.


- **Code Splitting và Code Insertion (Phân chia và chèn mã):** Phân chia mã nguồn thành các phần nhỏ hơn và chèn các phần mã mới vào, làm cho việc hiểu cấu trúc tổng thể trở nên phức tạp hơn.
- **Dummy Code Insertion (Chèn mã giả):** Thêm các đoạn mã không cần thiết hoặc không hoạt động vào mã nguồn để làm cho việc phân tích và đọc mã trở nên khó khăn hơn.
- **Transformations and Encoding (Biến đổi mã và mã hóa):** Thực hiện biến đổi, mã hóa (encoding) dữ liệu để làm cho thông tin trở nên khó hiểu và khó đọc.
- **Automated Tools (Sử dụng các công cụ):** Có các công cụ tự động có thể thực hiện các kỹ thuật obfuscation, từ việc thay đổi tên biến đến việc tái cấu trúc mã nguồn một cách tự động.

Mỗi kỹ thuật obfuscation có những ưu điểm và hạn chế riêng. Sự kết hợp của nhiều kỹ thuật này có thể tạo ra mã nguồn hoặc mã máy phức tạp và khó phân tích hơn đối với người đọc. Tuy nhiên, việc sử dụng obfuscation cũng có thể gây ra khó khăn cho việc debug và bảo trì mã nguồn.

#### 4 BIỂU MẪU/ PHỤ LỤC

STT	Biểu mẫu/ Phụ lục	Mã hiệu
1	Phụ lục Checklist Security API	PL01-HDCV-KCN-005



	<p style="text-align: center;"><b>PHỤ LỤC</b> <b>TÍCH HỢP STRINGER</b> <b>OBFUSCATOR</b></p>	<p>Mã hiệu: PL01-HDCV-KCN-005 Phiên bản: v1.0 Ngày ban hành: 01/12/2023</p>
---	--	---

Công cụ này có tác dụng obfuscate file jar khi build, dành cho team Java.

#### 4.1 Tải và cài đặt

- Đường link tải phần mềm sẽ được gửi cùng với mã kích hoạt qua email sau khi đăng ký sử dụng Stringer thành công tại [jfxstore.com](http://jfxstore.com).
- Để kích hoạt Stringer máy tính cần có kết nối internet.
- Sau khi tải bản cài đặt, giải nén và đi tới thư mục vừa giải nén chạy lệnh: `java -jar stringer.jar -activate` và làm theo hướng dẫn.

#### 4.2 Sử dụng CLI

Chạy lệnh

```
java -jar stringer.jar <options> <src> <dest>
```

Trong đó <src> là đường dẫn file jar gốc cần obfuscate, <dest> là đường dẫn file jar sau khi obfuscate.

Danh sách các options:

- `activate`: Kích hoạt Stringer
- `info`: Kiểm tra thông tin license
- `licenseFile`: Đường dẫn đến file license
- `configFile`: Đường dẫn đến file config

#### 4.3 Tích hợp với Ant

Thêm thẻ vào file build.xml

```
<taskdef name="stringer" classname="com.licel.stringer.AntTask"
classpath="${stringer.home}/stringer.jar" />
```

Cấu hình ant task

```
<stringer srcFile="${src.jar}" destFile="${dest.jar}"/>
```

hoặc

```
<stringer srcFile="${src.jar}" configFile="${stringer.config}"
destFile="${dest.jar}"/>
```

Ví dụ:

```
<stringer srcFile="${src.jar}" destFile="${dest.jar}">
  <verbose>true</verbose>
  <protectionElements>
    <protectionElement>
      <stringEncryption>
        <mode>filter</mode>
        <checkCaller>true</checkCaller>
        <filters>
          <filter>glob:com/sample/Library/**</filter>
        </filters>
      </stringEncryption>
    </protectionElement>
  </protectionElements>
</configuration>
```

#### 4.4 Tích hợp với Maven

Cài đặt các gói pom sau:

```
mvn install:install-file -Dfile=$STRINGER_HOME/stringer.jar -
DpomFile=$STRINGER_HOME/stringer.pom
mvn install:install-file -Dfile=$STRINGER_HOME/stringer-maven-plugin.jar -
DpomFile=$STRINGER_HOME/stringer-maven-plugin.pom
mvn install:install-file -Dfile=$STRINGER_HOME/stringer-annotations.jar -
DpomFile=$STRINGER_HOME/stringer-annotations.pom
```

Thêm thẻ plugin vào file pom.xml:

```
<plugin>
  <!-- STRINGER MUST BE THE LAST ONE IN A BYTECODE TRANSFORMATION CHAIN,
  I.E. THE LAST ONE BEFORE SIGNING PLUGINS IF THERE ARE ANY -->
  <groupId>com.licel</groupId>
  <artifactId>stringer-maven-plugin</artifactId>
  <version>${stringer.version}</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>stringer</goal>
      </goals>
      <configuration>
        <configFile>${stringer.config}</configFile>
```

```

        </configuration>
    </execution>
</executions>
</plugin>

```

#### 4.5 Tích hợp với Gradle

Thêm cấu hình sau vào file `gradle.build`

```

buildscript {
    repositories {
        jcenter()
        flatDir dirs: '../..'
    }

    dependencies {
        classpath ":stringer-gradle-plugin"
        classpath ":stringer"
    }

    apply plugin: "java"
    apply plugin: "stringer"

    stringer {
        configFile "${projectDir}/stringer.xml"
    }
}

```

#### 4.6 Tài liệu tham khảo thêm

<https://jfxstore.com/stringer/docs>