Trường Đại học Công nghệ Thông tin – ĐHQG Tp. HCM
Khoa Mạng máy tính và truyền thông

BÁO CÁO TỔNG KẾT ĐỒ ÁN MÔN HỌC

Môn học: Cơ chế hoạt động của mã độc

Tên chủ đề: Java WebShell Detection

Mã nhóm: CK14 Mã đề tài: S27

Lớp: NT230.P21.ANTT

1. THÔNG TIN THÀNH VIÊN NHÓM:

(Sinh viên liệt kê tất cả các thành viên trong nhóm)

STT	Họ và tên	MSSV	Email
1	Đinh Bạch Kiều Phương	21520406	21520406@gm.uit.edu.vn
2	Nguyễn Phúc Nhi	22521041	22521041@gm.uit.edu.vn
3	Phạm Trần Hồng Phúc	22521138	22521138@gm.uit.edu.vn

2. TÓM TẮT NỘI DUNG THỰC HIỆN:[1]

A. Chủ đề nghiên cứu trong lĩnh vực Mã độc: *(chọn nội dung tương ứng bên dưới)*

☐ Dev Track

☑ Research Track

B. Tên đề tài

Java WebShell Detection

C. Liên kết lưu trữ mã nguồn của nhóm:
Mã nguồn của đề tài đồ án được lưu tại:
https://drive.google.com/drive/folders/1jetlTWJizodq-il6L0Yp9wpEXdvXbFOZ

D. Tên tài liệu tham khảo chính:

Yu, X., Meng, W., Liu, Y., & Zhou, F. (2024). TridentShell: An enhanced covert and scalable backdoor injection attack on web applications. Journal of Network and Computer Applications, 223, 103823.

E. Tóm tắt nội dung chính:

[1] Ghi nội dung tương ứng theo mô tả

Trong bối cảnh các cuộc tấn công mạng ngày càng gia tăng, mã độc dạng web shell – đặc biệt là các backdoor được nhúng vào máy chủ web – trở thành một mối đe dọa nghiêm trọng đối với an toàn hệ thống. Bài báo giới thiệu TridentShell, một kiểu tấn công web backdoor hoàn toàn mới, có khả năng ẩn mình, khó truy vết, không để lại dấu vết file (fileless) và có thể vượt qua các cơ chế phát hiện mã độc tĩnh hiện có.

Khác với các phương pháp cũ sử dụng mã hóa hoặc làm rối mã để qua mặt hệ thống phòng thủ, TridentShell khai thác kỹ thuật Java Bytecode Instrumentation để chèn mã độc trực tiếp vào bộ nhớ của máy chủ ứng dụng Java như Tomcat, JBoss, WebLogic,... mà không làm thay đổi mã nguồn hay cấu hình hệ thống. Nhờ vậy, mã độc không bị phát hiện bởi các công cụ antivirus hay hệ thống kiểm tra mã nguồn truyền thống.

Một điểm nổi bật khác là TridentShell sử dụng mạng blockchain NKN làm kênh điều khiển và liên lạc (C&C server). Nhờ đó, việc liên lạc giữa attacker và máy chủ bị kiểm soát trở nên ẩn danh, phân tán, giúp tránh bị truy vết hoặc phân tích hành vi mạng. Thêm vào đó, mã độc còn có khả năng tự xóa dấu vết trong bộ nhớ bằng cách giải phóng file handle trên Windows, tránh bị lưu lại nhật ký hoạt động.

Bài báo đã thực hiện các thí nghiệm trên 5 nền tảng Java Web Server phổ biến và so sánh với các kỹ thuật cũ như China Chopper. Kết quả cho thấy TridentShell có khả năng lẩn tránh cực kỳ cao, vượt qua static detection, access control và phần mềm diệt virus như Windows Defender hoặc VirusTotal.

Tóm lại, TridentShell là một minh chứng cho sự nguy hiểm ngày càng tinh vi của các dạng mã độc mới, đồng thời đặt ra thách thức lớn đối với các phương pháp phòng thủ truyền thống. Bài nghiên cứu cũng đề xuất hướng phát hiện mới thông qua phân tích bytecode trực tiếp từ JVM bằng công cụ như HotSpot SA.

F. Tóm tắt các kỹ thuật chính được mô tả sử dụng trong đề tài:

Kỹ thuật 01: Java Bytecode Instrumentation

- Vai trò: Dùng để chèn mã độc vào bộ nhớ JVM của các máy chủ ứng dụng Java (Tomcat, JBoss, WebLogic...).
- Nhiệm vụ:
  - Hook các class xử lý HTTP (ví dụ: ApplicationFilterChain).
  - Thay thế class gốc bằng class chứa mã backdoor.
  - Không cần ghi file, không chỉnh sửa mã nguồn hoặc cấu hình.

➔ Giúp mã độc ẩn mình hiệu quả và khó bị phát hiện.

Kỹ thuật 02: Giao tiếp C&C qua mạng Blockchain NKN

- Vai trò: Thiết lập kênh điều khiển từ xa ẩn danh và phân tán giữa hacker và web shell.
- Nhiệm vụ:
  - Gửi/nhận lệnh (như whoami, calc) qua kênh NKN.
  - Không hiển thị gì trên trang web, không để lại log HTTP.

➔ Giúp tránh bị phát hiện bởi các hệ thống giám sát mạng hoặc nhật ký hệ thống.

Kỹ thuật 03: Gỡ dấu vết tấn công bằng kỹ thuật giải phóng file handle (Windows)

- Vai trò: Xóa sạch dấu vết của mã độc Load Module khỏi đĩa sau khi tấn công thành công.
- Nhiệm vụ:
  - Chiếm quyền debug (SE_DEBUG_NAME) để truy cập tiến trình hệ thống.
  - Tìm và giải phóng file handle của file .jar đã được load.

➔ Giúp đạt được mục tiêu tấn công không để lại dấu vết (fileless).

G. Môi trường thực nghiệm của đề tài:

Môi trường thực nghiệm của nhóm:

- Cấu hình máy tính: AMD Ryzen 7 - 5800H, 16GB RAM
- Các công cụ hỗ trợ sẵn có:
  - Eclipse IDE: dùng để lập trình, biên dịch và đóng gói JAR
  - Maven: quản lý dependencies.
- Ngôn ngữ lập trình để hiện thực phương pháp: Java 22.0.2
- Đối tượng nghiên cứu: Server Tomcat
- Tiêu chí đánh giá tính hiệu quả của phương pháp: chứng minh được phương pháp thực nghiệm của nhóm có thể tạo một backdoor trên Server. Sử dụng VirusTotal để kiểm tra xem đoạn mã có được phát hiện.

H. Kết quả đạt được: Công việc/tính năng/kỹ thuật mà nhóm thực hiện lập trình và triển khai cho demo:

Các công việc mà nhóm đã thực hiện:

- Nghiên cứu kỹ thuật Java Instrumentation mà tác giả đề cập để có thể tạo một backdoor lên trang web thử nghiệm sử dụng server Tomcat.
- Tạo một file JAR có khả năng chèn mã độc vào thời điểm chạy trong một web java thông qua thư viện Java.lang.Instrument và javassist.

- Sử dụng kỹ thuật trên, để hook mã độc vào class cụ thể bên trong ứng dụng web java qua cơ chế tự động load JAR của Tomcat.

Kết quả thực nghiệm của bài báo và của nhóm:

- Kết quả của bài báo:
  - TridentShellAgent có khả năng tấn công vào năm loại máy chủ ứng dụng web Java (Tomcat, JBoss, WebLogic, Resin, Jetty).
  - TridentShellAgent có khả năng chống phát hiện khỏi các phương pháp phát hiện tĩnh, các chính sách kiểm soát truy cập, phần mềm diệt virus và Windows Defender.
- Kết quả của nhóm:
  - Thực nghiệm được kỹ thuật Java Instrument, tạo ra một backdoor trên ứng dụng web có khả năng thực hiện lệnh để liệt kê ra các thư mục trên Server.
  - Tệp mã độc có khả năng lẫn trốn qua tool scan virus và Windows Defender.
- Ưu điểm của kỹ thuật Java Instrumentation Bycode:
  - Không cần thay đổi mã nguồn gốc
  - Khó bị phát hiện hơn so với webshell truyền thống.
  - Có thể chèn vào backdoor trước khi ứng dụng chạy (permain agent)
- Nhược điểm của kỹ thuật Java Instrumentation Bycode:
  - Cần quyền truy cập vào môi trường thực thi Java (JVM)
  - Yêu cầu hiểu viết về Java Bycode, dễ bị crash ứng dụng nếu cấu hình sai.

Công việc nhóm đã thực hiện và kết quả của chúng trong việc thực hiện phương pháp:

- Xây dựng ứng dụng web tượng trưng sử dụng server Tomcat để tượng trưng cho môi trường thực nghiệm.
- Viết một file mã độc có khả năng tạo một backdoor và chèn vào thời điểm khởi động của trang web bằng kỹ thuật Java Instumentation.

I. Các khó khăn, thách thức hiện tại khi thực hiện:

Trong quá trình thực hiện phương pháp trên, nhóm đã gặp một số khó khăn như sau:

- Chưa thực nghiệm được trên nhiều loại máy chủ ứng dụng web: Bài báo thực hiện kỹ thuật trên năm loại máy chủ ứng dụng web Java và đều thu được kết quả thành công. Do thời gian hạn chế, nên nhóm chỉ có thể nghiên cứu và thực hiện trên một máy chủ ứng dụng web là Tomcat.

- Chưa thực nghiệm được phiên bản cải tiến của TridentShellAgent: ở phiên bản cải tiến này, tác giả sử dụng mạng NKN (New Kind of Network) để liên lạc giữa attacker và backdoor. Kỹ thuật này chưa được đề cập chi tiết trong bài báo cũng như ít tài liệu tham khảo nên nhóm chưa hiểu được phương pháp thực hiện của phiên bản cải tiến này.
- Chưa thực nghiệm được cách phát hiện mã độc dựa trên phương pháp của tác giả: Do thời gian hạn chế, nhóm chưa triển khai được cách phát hiện của mã độc này.

3. TỰ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH SO VỚI KẾ HOẠCH THỰC HIỆN:

Mức độ hoàn thành: 70%

- Tạo ra được mã độc bằng phương pháp được đề cập trong bài báo
- Tuy nhiên: Chưa thực nghiệm được phiên bản cải tiến cũng như cách phát hiện của mã độc.

4. NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:

STT	Công việc	Phân công nhiệm vụ
1	Tìm hiểu về các khái niệm cơ bản	Đinh Bạch Kiều Phương
2	Tìm hiểu về phương pháp của đề tài	Nguyễn Phúc Nhi
3	Tìm hiểu, cài đặt các công cụ, thư viện và triển khai đề tài	Cả nhóm
5	Triển khai thực nghiệm và quay video demo	Cả nhóm
6	Tìm kiếm dataset có liên quan đến đề tài	Phạm Trần Hồng Phúc
7	Viết Report	Đinh Bạch Kiều Phương, Phạm Trần Hồng Phúc,
8	Làm Slide	Nguyễn Phúc Nhi, Đinh Bạch Kiều Phương
9	Thiết kế Poster	Phạm Trần Hồng Phúc, Nguyễn Phúc Nhi

BÁO CÁO TỔNG KẾT CHI TIẾT

Phần bên dưới của báo cáo này là tài liệu báo cáo tổng kết - chi tiết của nhóm thực hiện cho đề tài này.

Qui định: Mô tả các bước thực hiện/ Phương pháp thực hiện/Nội dung tìm hiểu (Ảnh chụp màn hình, số liệu thống kê trong bảng biểu, có giải thích)

A. Phương pháp thực hiện

Kiến trúc, thành phần của hệ thống trong bài báo

Bài báo đề xuất một hệ thống backdoor tàng hình có tên TridentShell, bao gồm 2 thành phần chính và một kênh điều khiển C&C, hoạt động theo mô hình chia lớp như sau:

1. **Load Module**

- Là thành phần đầu tiên được thực thi.
- Có nhiệm vụ kết nối với JVM đang chạy của server mục tiêu (thông qua Java Attach API).
- Gửi Agent Module vào bộ nhớ của JVM thông qua phương thức loadAgent().
- Sau khi hoàn tất nhiệm vụ, Load Module sẽ bị xóa khỏi hệ thống để không để lại dấu vết.

2. **Agent Module**

- Là thành phần chính chứa mã backdoor.
- Sử dụng Java Instrumentation API để hook vào các lớp xử lý HTTP trong JVM (như ApplicationFilterChain).
- Thay thế bytecode của các class mục tiêu để thêm khả năng thực thi lệnh (cmd) khi nhận đúng tham số password.
- Có thể khởi tạo một NKN client để giao tiếp với attacker.

3. **Kênh điều khiển C&C (Command & Control)**

- Sử dụng mạng blockchain NKN (New Kind of Network) làm kênh giao tiếp ẩn danh.
- Dữ liệu được mã hóa đầu-cuối (E2E), mỗi session sử dụng một channel khác nhau.
- TridentShell không trả kết quả ra giao diện HTTP mà gửi trực tiếp qua kênh C&C.

➔ Luồng hoạt động chính:
- Attacker triển khai Load Module → JVM bị điều khiển.
- Load Module đẩy Agent vào JVM → Hook lớp HTTP xử lý request.
- Attacker gửi request HTTP chứa password + cmd → Agent xử lý → thực thi lệnh.
- Kết quả lệnh được gửi ngược về attacker thông qua mạng NKN.
- Load Module bị xóa để đảm bảo không còn dấu vết file.

Kiến trúc, thành phần đã thực hiện (nội dung mà nhóm đã thực hiện)

Nhóm đã triển khai một phiên bản đơn giản hóa của hệ thống TridentShell, tập trung vào môi trường thực nghiệm với server Tomcat:

- Nghiên cứu và áp dụng kỹ thuật Java Instrumentation nhằm chèn mã độc vào runtime của ứng dụng Java.
- Viết một file JAR sử dụng thư viện java.lang.instrument và javassist để thực hiện thay đổi class tại thời điểm chạy.
- Hook mã độc vào một class cụ thể trong ứng dụng web thông qua cơ chế tự động nạp JAR của Tomcat.
- Tạo một ứng dụng web giả lập để kiểm chứng khả năng thực thi lệnh từ backdoor đã được chèn.

B. Chi tiết cài đặt, hiện thực

<cách cài đặt, lập trình trên máy tính, cấu hình máy tính sử dụng, chuẩn bị dữ liệu, v.v>

- Chuẩn bị các thư mục liên quan bao gồm:
  - Thư mục MyServletProject: chứa các file liên quan để build một trang web thử nghiệm sử dụng máy chủ Tomcat. Với giao diện Login như hình.
  - Server Tomcat: phiên bản v10.0
  - Thư mục TridentShellAgent: chứa source code của mã độc dùng để tạo backdoor.
- Chuẩn bị Eclipse IDE để thực hiện build mã độc (.JAR) cũng như là chạy server web.

- Cấu hình máy tính: AMD Ryzen 7 - 5800H, 16GB RAM

1. Xây dựng mã độc (TridentShellAgent):

- Tạo một dự án Java mới trong Eclipse cho mã độc.
- Thêm các thư viện phụ thuộc thông qua tệp pom.xml:
  - **Javassist**: Dùng để thao tác bytecode, cho phép chèn mã backdoor vào các lớp mục tiêu.
  - **JNA và JNA-Platform**: Hỗ trợ truy cập native trên Windows để thực hiện cơ chế dọn dẹp tệp JAR.

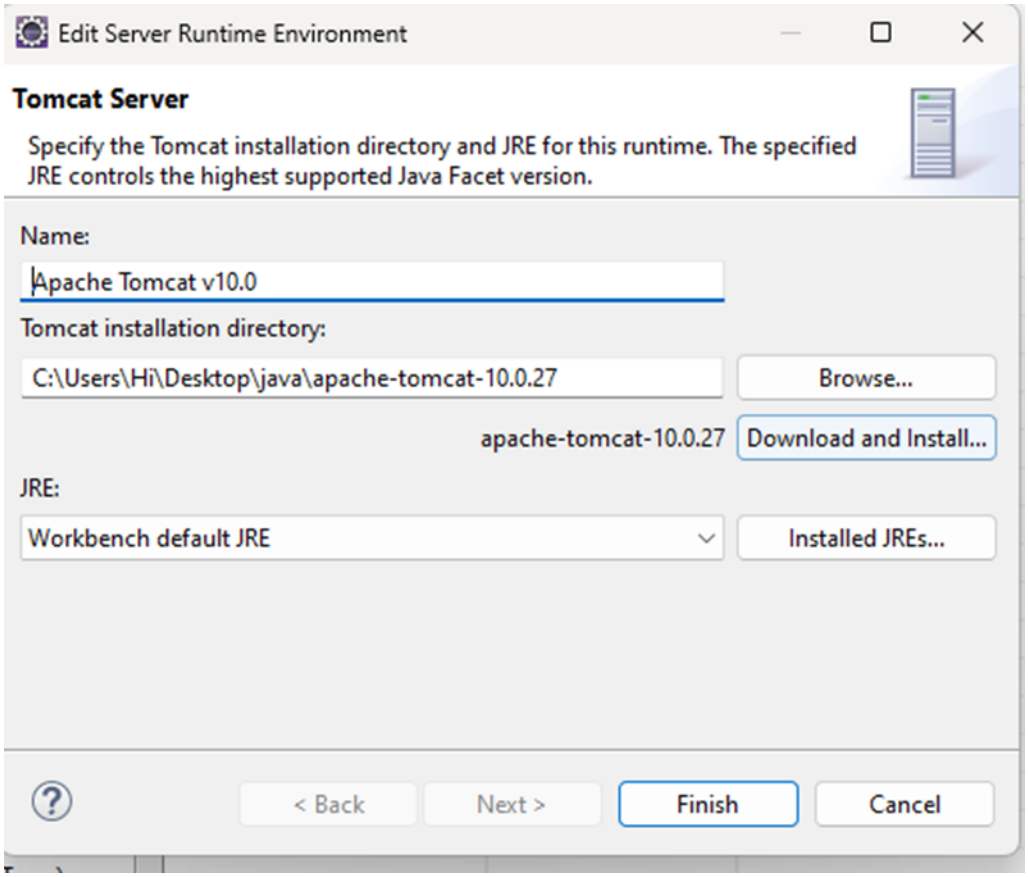
```
<dependencies>
    <dependency>
        <groupId>org.javassist</groupId>
        <artifactId>javassist</artifactId>
        <version>3.29.2-GA</version>
    </dependency>
    <dependency>
        <groupId>net.java.dev.jna</groupId>
        <artifactId>jna</artifactId>
        <version>5.14.0</version>
    </dependency>
    <dependency>
        <groupId>net.java.dev.jna</groupId>
        <artifactId>jna-platform</artifactId>
        <version>5.14.0</version>
    </dependency>
</dependencies>
```

- Sử dụng plugin maven-assembly-plugin để xây dựng tệp JAR (TridentShellAgent-1.0-SNAPSHOT-jar-with-dependencies.jar) bao gồm tất cả các phụ thuộc.
- Cấu hình tệp manifest trong JAR để chỉ định lớp premain là com.da_uit.agent.TridentShellAgent, cho phép mã độc hoạt động như một Java agent.

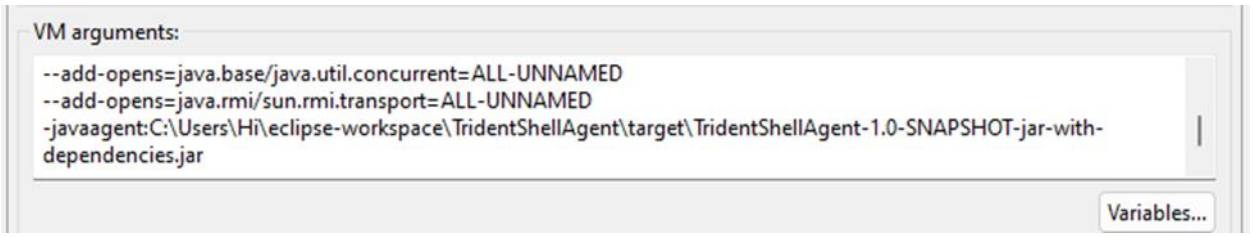
```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>3.3.0</version>
            <configuration>
                <archive>
                    <manifestEntries>
                        <Premain-Class>com.da_uit.agent.TridentShellAgent</Premain-Class>
                        <Can-Redefine-Classes>true</Can-Redefine-Classes>
                        <Can-Retransform-Classes>true</Can-Retransform-Classes>
                    </manifestEntries>
                </archive>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
            </configuration>
            <executions>
                <execution>
                    <id>make-assembly</id>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

2. Xây dựng ứng dụng web (MySevletProject):

- Tạo một dự án web trong Eclipse, bao gồm servlet AuthServlet.java để xử lý đăng nhập và đăng ký người dùng.
- Tích hợp máy chủ Tomcat v10.0 vào Eclipse để triển khai ứng dụng.



- Thêm tùy chọn -javaagent:<đường dẫn đến TridentShellAgent.jar> vào tham số JVM trong cấu hình Tomcat, nhằm tải mã độc khi máy chủ khởi động.



3. Triển khai và chạy:

- Khởi chạy mã độc:

```
[INFO] -----------------< com.da_uit.agent:TridentShellAgent >-----------------
[INFO] Building TridentShellAgent 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ TridentShellAgent ---
[INFO] Deleting C:\Users\Hi\eclipse-workspace\TridentShellAgent\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ TridentShellAgent ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ TridentShellAgent ---
[INFO] Recompiling the module because of changed source code.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file with javac [debug target 1.8] to target\classes
[WARNING] bootstrap class path is not set in conjunction with -source 8
  not setting the bootstrap class path may lead to class files that cannot run on JDK 8
    --release 8 is recommended instead of -source 8 -target 1.8 because it sets the bootstrap class path automatically
[WARNING] source value 8 is obsolete and will be removed in a future release
[WARNING] target value 8 is obsolete and will be removed in a future release
[WARNING] To suppress warnings about obsolete options, use -Xlint:-options.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ TridentShellAgent ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource from src\test\resources to target\test-classes
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ TridentShellAgent ---
[INFO] Recompiling the module because of changed dependency.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ TridentShellAgent ---
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ TridentShellAgent ---
[INFO] Building jar: C:\Users\Hi\eclipse-workspace\TridentShellAgent\target\TridentShellAgent-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- assembly:3.3.0:single (make-assembly) @ TridentShellAgent ---
[INFO] Building jar: C:\Users\Hi\eclipse-workspace\TridentShellAgent\target\TridentShellAgent-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.396 s
[INFO] Finished at: 2025-06-04T09:16:43+07:00
[INFO] ------------------------------------------------------------------------
```

- Khởi động Tomcat từ Eclipse để chạy ứng dụng web:

```
Jun 04, 2025 9:17:37 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Server version name:   Apache Tomcat/10.0.27
Jun 04, 2025 9:17:37 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Server built:          Oct 3 2022 14:18:31 UTC
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Server version number: 10.0.27.0
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: OS Name:               Windows 11
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: OS Version:            10.0
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Architecture:          amd64
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Java Home:             C:\Program Files\Java\jdk-23
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: JVM Version:           23.0.1+11-39
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: JVM Vendor:            Oracle Corporation
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: CATALINA_BASE:         C:\Users\Hi\eclipse-workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp1
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: CATALINA_HOME:         C:\Users\Hi\Desktop\java\apache-tomcat-10.0.27
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: -Dcatalina.base=C:\Users\Hi\eclipse-workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp1
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: -Dcatalina.home=C:\Users\Hi\Desktop\java\apache-tomcat-10.0.27
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: -Dwtp.deploy=C:\Users\Hi\eclipse-workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp1\wtpwebapps
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: -Djava.library.path=C:\Users\Hi\Desktop\java\apache-tomcat-10.0.27\bin
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: --add-opens=java.base/java.lang=ALL-UNNAMED
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: --add-opens=java.base/java.io=ALL-UNNAMED
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: --add-opens=java.base/java.util=ALL-UNNAMED
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: --add-opens=java.base/java.util.concurrent=ALL-UNNAMED
Jun 04, 2025 9:17:38 AM org.apache.catalina.startup.VersionLoggerListener log
INFO: Command line argument: --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
```

- Truy cập giao diện đăng nhập của ứng dụng để kiểm tra chức năng thông thường.
- Gửi các yêu cầu HTTP chứa tham số đặc biệt (secretKey=backdoor và cmd=<lệnh>) để kích hoạt backdoor và thực thi lệnh hệ thống.

```
C:\Users\Hi>curl -X POST http://localhost:8080/UserAuthWeb/AuthServlet -d "secretKey=backdoor&cmd=dir"
```

C. Kết quả thực nghiệm

<mô tả hình ảnh về thực nghiệm, bảng biểu số liệu thống kê từ thực nghiệm, nhận xét về kết quả thu được.>

Mô Tả Thực Nghiệm

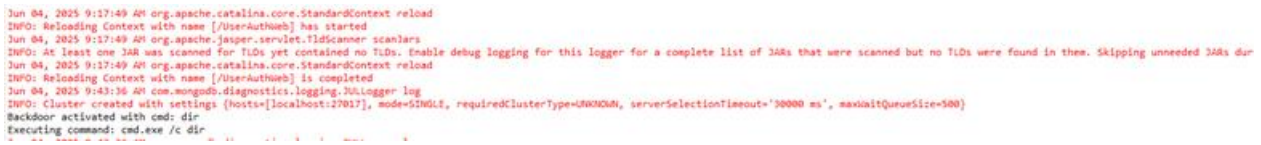
- **Môi Trường Thử Nghiệm:**
  - Ứng dụng web được triển khai trên máy chủ **Tomcat v10.0**.
  - Mã độc (TridentShellAgent) được tải thông qua tùy chọn -javaagent khi khởi động Tomcat.
- **Các Bước Thử Nghiệm:**
  - Truy cập trang đăng nhập của ứng dụng web để xác nhận giao diện và chức năng hoạt động bình thường.
  - Gửi yêu cầu HTTP chứa tham số backdoor để thực thi lệnh hệ thống. Ví dụ:
  - Kiểm tra nhật ký (log) của máy chủ để xác nhận:
    - Việc chuyển đổi (transform) các lớp mục tiêu thành công.
    - Backdoor được kích hoạt và thực thi lệnh.

Kết Quả Thu Được

- **Chuyển Đổi Lớp (Class Transformation):**
  - Mã độc đã thành công trong việc chèn backdoor vào các lớp mục tiêu.
  - Nhật ký hiển thị thông báo:

```
Jun 04, 2025 9:17:37 AM com.da_uit.agent.TridentShellAgent premain
INFO: Agent loaded successfully
```

- **Kích Hoạt Backdoor:**
  - Yêu cầu HTTP với tham số secretKey=backdoor đã kích hoạt backdoor và thực thi lệnh hệ thống thành công.



- **Dọn Dẹp Tệp JAR:**
  - Trên hệ điều hành Windows, mã độc đã cố gắng xóa tệp JAR sau khi được tải.
  - Nhật ký hiển thị

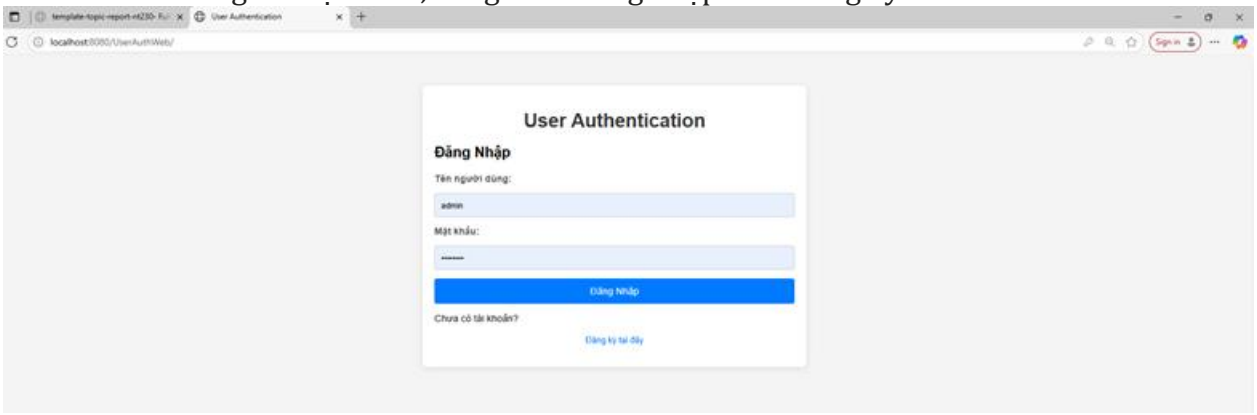
```
Jun 04, 2025 9:17:37 AM com.da_uit.agent.TridentShellAgent cleanupAgentJar
```

Nhận Xét Về Kết Quả

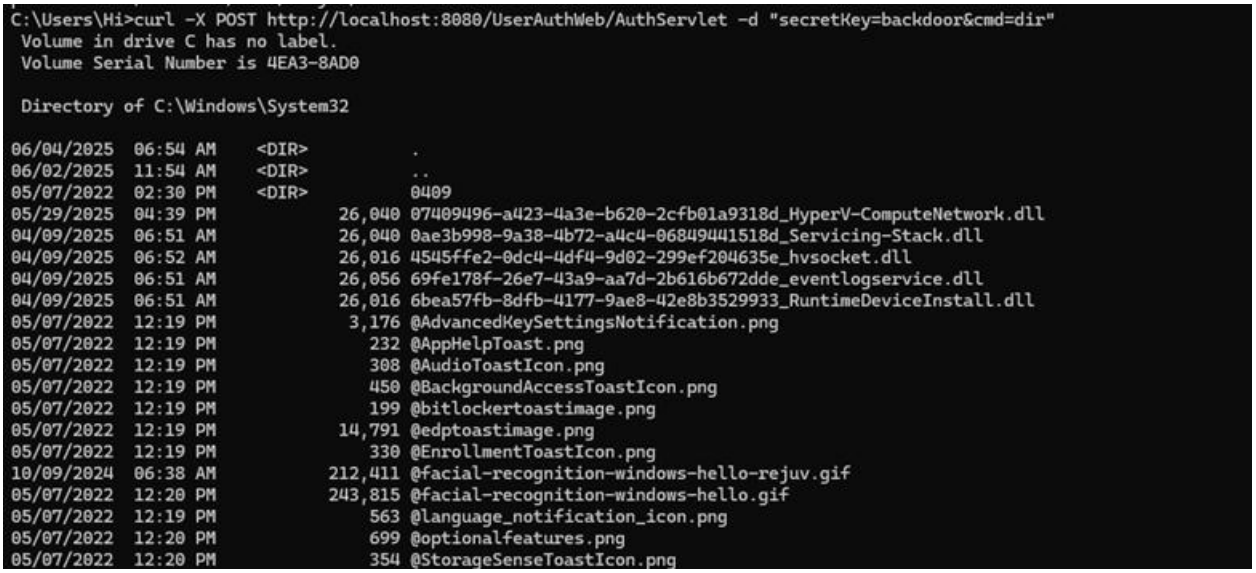
- **Hiệu Quả của Backdoor:**
  - Backdoor hoạt động đúng như mong đợi, cho phép thực thi lệnh hệ thống từ xa thông qua yêu cầu HTTP mà không làm gián đoạn chức năng chính của ứng dụng web (đăng nhập/đăng ký).
- **Tính Tương Thích:**
  - Mặc dù mã độc được thiết kế để nhắm vào nhiều framework web (Tomcat, Resin, Jetty, WebLogic), trong thử nghiệm này chỉ tập trung vào Tomcat v10.0 và hoạt động hiệu quả.
- **Cơ Chế Dọn Dẹp:**
  - Cơ chế dọn dẹp tệp JAR trên Windows thành công nhờ sử dụng JNA để mở khóa quyền debug và giải phóng tệp đang được sử dụng, giúp giảm dấu vết của mã độc.
- **Hạn Chế và Vấn Đề:**
  - Backdoor có thể bị phát hiện nếu hệ thống giám sát các tham số bất thường trong yêu cầu HTTP (như secretKey=backdoor).
  - Việc chèn mã thông qua thao tác bytecode có thể bị phát hiện bởi các công cụ bảo mật kiểm tra tính toàn vẹn của lớp.

Hình Ảnh Thực Nghiệm

- **Hình 1: Giao diện Đăng Nhập**
  - Hiển thị trang đăng nhập của ứng dụng web với các trường nhập tên người dùng và mật khẩu, cùng nút "Đăng nhập" và "Đăng ký".



- **Hình 2: Kết Quả Thực Thi Lệnh**
  - Hiển thị kết quả của lệnh dir khi gửi yêu cầu:



Bảng Thống Kê

Thử Nghiệm	Kết Quả
Chuyển đổi lớp	Thành công cho ApplicationFilterChain
Kích hoạt backdoor	Thành công, lệnh thực thi và trả về kết quả
Dọn dẹp tệp JAR	Thành công trên Windows

D. Hướng phát triển

Hướng phát triển tiềm năng:

- Tự động hóa toàn bộ quá trình tấn công thông qua giao diện điều khiển hoặc tập lệnh, giúp kiểm thử bảo mật dễ dàng hơn trong môi trường lab.
- Tích hợp sâu với mạng NKN thực tế, từ đó đánh giá khả năng ẩn danh và truyền dữ liệu qua các kênh blockchain trong các môi trường thật thay vì mô phỏng.
- Mở rộng đối tượng tấn công sang các nền tảng phi Java như PHP, .NET, sử dụng cơ chế tương tự như injection vào runtime.
- Xây dựng công cụ phát hiện TridentShell tự động, kết hợp với hệ thống giám sát bytecode hoặc phân tích hành vi trong thời gian thực (real-time JVM inspection).
- Phân tích đối kháng (adversarial analysis) để xây dựng hệ thống phòng thủ chủ động đối với các biến thể fileless malware.

Nhận xét về tính ứng dụng:

- TridentShell là mô hình tấn công nâng cao (APT-like) có tính ẩn danh và khó truy vết cao, phù hợp để dùng trong nghiên cứu tấn công thử nghiệm (red team) hoặc xây dựng bài lab kiểm thử an toàn.

- Mặc dù nguy hiểm nếu bị lạm dụng, nhưng đề tài mang lại giá trị lớn cho việc đánh giá khả năng phát hiện của các công cụ bảo mật hiện có, từ đó cải thiện giải pháp phòng thủ.
- Do tận dụng cơ chế Instrumentation hợp lệ trong Java, kỹ thuật có thể ứng dụng để phát triển công cụ giám sát hệ thống hoặc kiểm thử mã độc hợp pháp trong môi trường nghiên cứu.

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
- Đặt tên theo định dạng: [Mã lớp]-Project_Final_NhomX_Madetai. (trong đó X và Madetai là mã số thứ tự nhóm và Mã đề tài trong danh sách đăng ký nhóm đồ án).

  Ví dụ: [NT521.N11.ANTT]-Project_Final_Nhom03_CK01.
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT