Solvinity
Secure Managed IT Services

Informatie Log4J a.k.a. 'Log4Shell'

Referentie: CVE-2021-44228

Door: Marc Guardiola, CISO Solvinity

1. Disclaimer

Onderstaande informatie is op basis van wat wij (Solvinity) nu weten. De situatie verandert regelmatig, de informatie van nu kan over een paar uur alweer achterhaald zijn. We proberen deze informatie zo goed als mogelijk correct, volledig en up-to-date te houden, maar kunnen geen garanties geven.

2. Versiebeheer

Versie	Datum
1.0	13-12-2021
Initiële versie	

Versie	Datum
1.1	17-12-2021
• Informatie toegevoegd over gerelateerde kwetsbaarheden • Verdere verdieping in kwetsbare en niet-kwetsbare Log4j versies • Update op workarounds	

Versie	Datum
1.2	19-12-2021

In hoofdstuk 6 en 7 van dit document stond eerder al informatie over de Log4j 2.x kwetsbaarheid CVE-2021-45046. Dit is niet de origineel ontdekte ‘Log4Shell’ kwetsbaarheid, maar valt als extra ‘attack vector’ wel onder de Log4Shell paraplu. De potentiële impact van deze kwetsbaarheid CVE-2021-45046 is groter dan gedacht. De CVSS score, een standaard om het risico te bepalen, is daarom verhoogd van 3.7 naar 9.0. Eerder dacht men dat het via deze kwetsbaarheid alleen mogelijk was om de applicatie te laten crashen (Denial of Service), maar nu blijkt dat deze kwetsbaarheid in bepaalde gevallen ook kan leiden tot Remote Code Execution. Vooralsnog lijkt het er wel op dat lang niet elke applicatie hier vatbaar voor is. Log4j 2.16 of hoger, en Log4j 2.12.2 mitigeren deze kwetsbaarheid. De workaround, het verwijderen van de JndiLookup class mitigeert het risico ook.

Er is ook een nieuwe Log4j 2.x kwetsbaarheid ontdekt, CVE-2021-45105 met een CVSS score van 7.5 . Afhankelijk van wat er wordt gelogd is het mogelijk om de applicatie te laten crashen (Denial of Service). Dat is natuurlijk vervelend, maar staat qua risico niet in verhouding tot kwetsbaarheden waarbij code kan worden uitgevoerd. Deze kwetsbaarheid is opgelost in Log4j versie 2.17 (voor Java 8 of hoger). Voor Java 7 is nog geen update. Volgens de beschikbare informatie mitigeert de workaround, het verwijderen van de JndiLookup class, dit risico niet.

Deze nieuwe informatie is verwerkt in hoofdstuk 6 en 7 van dit document.

3. Introductie

Deze informatie is bedoeld voor iedereen die meer wil weten over de kwetsbaarheden in Log4j. Enige technische achtergrond helpt om bepaalde detailinformatie beter te begrijpen, maar is niet strikt noodzakelijk. Het doel van deze informatie is om mensen een beter begrip te geven over wat er nu precies aan de hand is, welk risico dit oplevert en op welke manier dat risico kan worden gemitigeerd. Deze informatie heeft een algemeen karakter en gaat niet specifiek in op applicaties onder beheer van Solvinity. We gaan in op:

1. Wat is Log4j
2. Kwetsbaarheid CVE-2021-44228, ook bekend als 'Log4Shell'
3. Kwetsbare en niet kwetsbare Log4j versies
4. Workarounds
5. Hoe nu verder

4. Wat is Log4j ?

Log4j is een veel gebruikte logging library binnen Java applicaties. Log4j is dus geen standalone applicatie, maar een library: een stuk software dat applicatiebouwers kunnen gebruiken voor een bepaalde functie van de applicatie. Applicatiebouwers hoeven die functionaliteit dan niet zelf te schrijven - over het algemeen een goed idee, en de kracht van Open Source. Dit brengt de kosten omlaag en de kwaliteit omhoog. Zoals in elk stuk software kan er ook in een library een kwetsbaarheid zitten, dat is hier helaas het geval. Dit is ook nog eens een zeer veel gebruikte library, een applicatie kan deze namelijk gebruiken om mee te loggen.

Elke applicatie logt (of zou dat moeten doen), de meest gebruikte library voor die functionaliteit is Log4j, dus vinden we deze library in ontzettend veel applicaties. Als je een Java applicatie hebt maar nog nooit hebt gehoord van log4j, heb je waarschijnlijk toch Log4j. Maar ook als je denkt dat je geen Java applicatie draait, zit je er waarschijnlijk naast. Java is echt overal. Ook als je een applicatie hebt die primair niet op Java draait, kan dat van een subcomponent van diezelfde applicatie weer anders zijn. Het toegenomen gebruik van kant-en-klare microservices (containers) maakt het dan niet altijd duidelijker. Veel (virtual) appliances draaien ‘onder water’ Java. Denk aan VPN gateways, firewalls, proxies, mailservers, IAM oplossingen, et cetera. Ook op je wificontroller kan Java draaien (zoals UniFi), maar denk ook aan andere smart devices. Het NCSC heeft een inventarisatie met veel gebruikte software gemaakt, deze is hier te vinden. Dit is echter geen 100% complete lijst, dat kan ook niet. Het is wel een goed startpunt om zelf een inventarisatie te maken en kan na handmatige inventarisaties worden gebruikt als een cross-check.

Het gaat niet alleen over serverapplicaties, maar ook over clients. Het populaire spel Minecraft heeft bijvoorbeeld een Java client - maar Minecraft servers draaien ook op Java. Wanneer eerst de server wordt aangevallen, kunnen op deze gecompromitteerde servers daarna de verbonden clients worden ‘misleid’ om het stuk tekst te loggen dat de code ophaalt. Minecraft is slechts een voorbeeld. Veel businessapplicaties hebben ook een Java client.

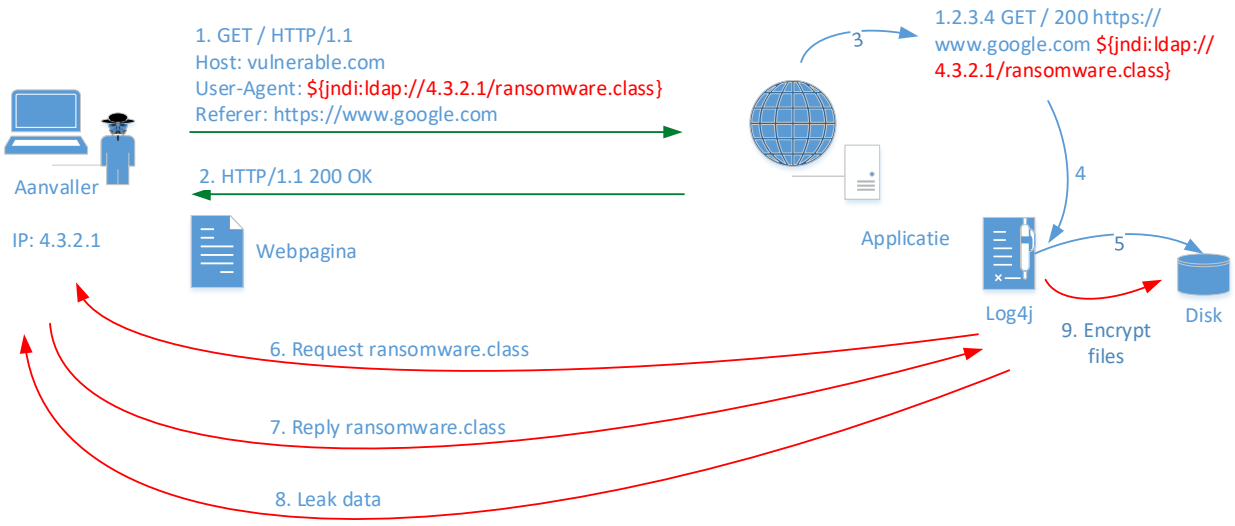
Hieronder gaan we in op hoe de aanval werkt, de verschillende Log4j versies, en welke acties moeten worden ondernomen.

5. Kwetsbaarheid CVE-2021-44228 'Log4Shell'

In Log4j zijn meerdere kwetsbaarheden ontdekt, zowel recent als in het verleden, maar één steekt er met kop en schouders boven uit: De kwetsbaarheid bekend onder CVE-2021-44228 of 'Log4Shell'. In het kort zorgt deze kwetsbaarheid ervoor dat, wanneer een applicatie via log4j een bepaalde tekst logt, er code wordt uitgevoerd op de server waar Log4j op draait. Deze tekst is dus eigenlijk een stukje code die log4j vervolgens oppakt en uitvoert. Dit type aanval staat bekend als 'code injection'.

De functionaliteit van het stuk code dat kan worden geïnjecteerd is gelimiteerd. Feitelijk beperkt dit zich tot het ophalen en uitvoeren van weer een ander stuk code. Echter, deze code kan worden opgehaald van een willekeurige locatie op het internet, de aanvaller kan de code zelf maken. Deze code is niet beperkt, en wordt dan door Log4j uitgevoerd met de rechten van de applicatie.

Onderstaand diagram geeft een idee van hoe het kan werken. Normaal gesproken eindigt het contact met de client (links) bij stap 2 en het logproces bij stap 5. Browsers sturen standaard ook de browser versie mee, door die te vervangen met de code zien we dat er meer communicatie plaatsvindt. Er wordt nog een stuk software opgehaald (stap 6 en 7). Deze software wordt uitgevoerd, vervolgens kunnen bestanden worden versleuteld (stap 8) en kan data worden gelekt (stap 9). Het diagram is enigszins versimpeld, in de praktijk zijn er een paar extra stappen en zijn er meer manieren om de code te injecteren.



Het is belangrijk om te realiseren dat de stappen 8 en 9 niet direct plaats hoeven te vinden. Vaak wachten aanvallers tot een voor de aanvaller gunstig moment, de feestdagen bijvoorbeeld. Het duurt dan vaak langer voor de aanval wordt gedetecteerd en de aangevallen organisatie actie heeft ondernomen om de impact te beperken. Het diagram laat niet elk scenario zien.

Tot slot is het belangrijk om te realiseren dat een server niet direct bereikbaar hoeft te zijn vanaf het internet. Zolang er maar, op wat voor manier dan ook, invloed kan worden uitgeoefend op wat er wordt gelogd. Later meer hierover.

Er zijn dus een aantal voorwaarden waar aan moet worden voldaan om de aanval te doen slagen.

- Een kwetsbare Log4j. Dit is best ingewikkeld, de uitleg staat iets verder in dit document.
- Invloed op wat er wordt gelogd
- Toegang naar een door de aanvaller gecontroleerd systeem
- Voldoende rechten

5.1 Invloed op wat er wordt gelogd

Invloed uitoefenen op wat er wordt gelogd kan op veel manieren, maar is afhankelijk van de applicatie. Hieronder een aantal voorbeelden hoe een aanvaller ervoor kan zorgen dat het stuk tekst wordt gelogd dat ervoor zorgt dat er code wordt opgehaald en uitgevoerd:

- Wanneer de applicatie bij het inloggen van een gebruiker de gebruikersnaam logt, kan een aanvaller de gebruikersnaam aanpassen.
- Wanneer de applicatie elk bezoek logt, kan de aanvaller de bezochte pagina aanpassen.
- Wanneer de browser wordt gelogd (de User-Agent) kan deze worden aangepast.
- Wanneer alleen illegale requests worden gelogd, kan de aanvaller expres een illegaal request uitvoeren waar deze tekst in voorkomt.
- In het geval van clients is het afhankelijk van de client. De client moet dan zelf connectie maken met een malafide (of reeds gehackte) server. Ook dit scenario is niet zo theoretisch als het misschien klinkt, een bekend voorbeeld is Minecraft, maar er zijn zeker meer voorbeelden te bedenken met businessapplicaties.

Om de kwetsbaarheid te kunnen misbruiken moet de server dus een specifiek stuk tekst loggen. De server hoeft hier niet direct voor benaderbaar te zijn vanaf een extern netwerk – indirect, via een reverse proxy en een webserver komt een verzoek vaak onaangepast op applicatieservers uit. Maar indirect betekent meer dan dat. Er bestaat bijvoorbeeld een scenario waarbij een applicatieserver op deze manier gecompromitteerd wordt. Systemen bereikbaar vanaf deze applicatieserver kunnen eveneensgecompromitteerd worden: direct door die applicatieserver zelf, of door een client van die applicatieserver iets te laten loggen. Maar er is nog een manier. Als de applicatieserver bijvoorbeeld orders wegschrijft die later worden uitgelezen door een billingsysteem, kan de aanvaller een order laten wegschrijven met het malafide stuk tekst erin. Als het billingsysteem dan de orders uitleest en dat deel logt, wordt deze ook geïnfecteerd. Logs worden vaak ook extern opgeslagen, soms direct en soms pas na een paar dagen. Ook dan kan de logstroom door een Log4j library van het logarchiveringssysteem heengaan. Dat klinkt misschien theoretisch maar dat is het niet. Scans die Solvinity heeft uitgevoerd, tonen soms pas na dagen een resultaat dat aantoont dat een applicatie, diep verstopt in een netwerk, kwetsbaar is.

5.2 Toegang naar het internet

Zoals gezegd is het belangrijkste lek vrij beperkt, namelijk het ophalen van nieuwe code. Om een door de aanvaller gekozen stuk code uit te voeren, moet de applicatieserver om die code op te halen ook toegang hebben naar het internet. Één open TCP-poort naar buiten is voldoende. Deze toegang is bij Solvinity standaard niet toegestaan. Dit betekent niet dat het helemaal nooit wordt opengezet, maar dat gebeurt slechts in zeldzame gevallen waarbij het echt nodig is voor de werking van de applicatie.

Er is nog een ander scenario waarin er toch informatie kan lekken: via DNS. Dit lijkt vooralsnog beperkt tot de inhoud van bepaalde operating system environment variables. Toch zou een creatieve aanvaller hier misschien meer mee kunnen dan dat we nu kunnen bedenken. Directe toegang naar buiten over DNS is niet nodig, dit kan ook lopen via een (vaak interne) DNS server. Daarnaast is het goed mogelijk om via DNS te controleren of een applicatie lek is, zelfs als deze niet zomaar naar buiten kan communiceren om de door de aanvaller gemaakte code op te halen. Dit kan voor een aanvaller een goede reden zijn om het toch te blijven proberen.

Werkplekken mogen vaker zelf connectie maken met het internet, afhankelijk van hoe de werkplekomgeving is ingeregeld.

5.3 Voldoende rechten

Een nachtmerriescenario is natuurlijk als de applicatie draait als ‘root’ of ‘administrator’. Op servers beheerd door Solvinity is dit niet het geval, dit is ook bepaald niet niet best-practice. Toch zijn de rechten van applicaties vaak genoeg om veel schade aan te richten. Een applicatie heeft bijvoorbeeld vaak toegang tot bestanden en/of een database – de aanvaller dan dus ook. Een applicatie mag ook processen starten, een van die processen zou een ransomware of een cryptominer kunnen zijn. In het geval van een ransomware kan er alleen data versleuteld of gelekt worden waar de applicatie rechten tot heeft. NB: Bij Solvinity hebben applicaties geen toegang tot backups.

Bij werkplekken is dit anders. Een Java client op een werkplek draait doorgaans met de rechten van de gebruiker, gebruikers hebben doorgaans weer toegang tot veel data. Hier geldt eveneens dat die werkplek dan met een Java client eerst verbinding moet maken met een malafide applicatie.

6. Kwetsbare en niet kwetsbare Log4j versies

Hierover gaan verschillende verhalen de ronde, de situatie is vrij complex. Allereerst: Log4j versie 1.x.

6.1 Log4j 1.x

Log4j versie 1.x is sinds 2015 end-of-life. Deze versie kent al verschillende vulnerabilities, en zou eigenlijk nergens meer mogen draaien. log4j versie 1.x is niet kwetsbaar voor CVE-2021-44228, al wordt de 'scope' van deze kwetsbaarheid nogal eens opgerekt waardoor dit beeld wat vertroebelt. Er zijn namelijk twee andere bekende kwetsbaarheden (een oude en een nieuwe), waarbij de meest recente wel wat raakvlakken heeft met CVE-2021-44228.

- CVE-2019-17571: Ook een remote code execution vulnerability. Bekend sinds 2019. Voor zover we weten niet 'zomaar' te misbruiken, er is minder over bekend en er moet aan condities worden voldaan die niet overal van toepassing zijn.
- CVE-2021-4104: Deze heeft een link met CVE-2021-44228, a.k.a. 'Log4Shell'. Het gaat dan om een aanvalsmethode die specifieke omstandigheden vereist om te kunnen slagen.

De laatstgenoemde kwetsbaarheid, CVE-2021-4104, wordt soms ook meegewogen bij statements dat Log4j ook vatbaar zou zijn voor Log4Shell.

Deze twee kwetsbaarheiden komen qua risico niet in de buurt van CVE-2021-44228. Dat heeft vooral te maken met de kans, niet de impact. De kans dat de manier waarop Log4j wordt gebruikt vatbaar is voor CVE-2019-17571 of CVE-2021-4104 is kleiner, maar als deze vatbaar is, kunnen de gevolgen even groot zijn.

In alle gevallen geldt dat Log4j 1.x al lang geleden had moeten worden geüpdate. Als dat niet is gebeurd, geldt dat waarschijnlijk voor nog veel meer libraries - dan is hier blijkbaar geen controle op. Het zou dan gek zijn om alleen Log4j versie 1.x te updaten en de rest niet te onderzoeken - en het proces te verbeteren. De kanttekening hierbij is dat Log4j veel aandacht in de media krijgt. Elk regeltje code in Log4j (inclusief versie 1.x) krijgt nu in één dag misschien meer aandacht dan de opgetelde totale aandacht sinds de eerste versie van Log4j (uitgebracht in 2001). Met dat gegeven is de kans vrij groot dat er nieuwe kwetsbaarheden worden gevonden, ook in Log4j 1.x. Updaten dus, al hoeft dat strikt genomen niet voor CVE-2021-44228 (Log4Shell). Let goed op dat je Java 7 of Java 8 draait. Als dit niet het geval is moet Java worden geüpdatet, wat gevolgen kan hebben voor de werking van de applicatie. Java 6 is echter ook al heel lang end of life, hopelijk vormt dat geen probleem.

6.2 Log4j 2.x

Hier wordt het wat ingewikkeld. Elke subversie vereist een andere versie van Java, daarmee is er een extra afhankelijkheid.

Log4j versie	Vereiste Java versie	Fix vanaf
2.0 t/m 2.3	Java 6	Geen
2.4 t/m 2.12	Java 7	2.12.2
2.13 en hoger	Java 8	2.15, 2.16, 2.17 (zie hieronder voor uitleg)

Voor Log4j versies op Java 6 is momenteel geen update beschikbaar, en voor Java 7 is er 2.12.2. Maar er zijn meerdere updates voor Java 8, hoe zit dat?

Net als in het vorige verhaal over Log4j 1.x, is ook voor Log4j 2.x een nieuwe kwetsbaarheid ontdekt met raakvlakken met 'Log4Shell', namelijk CVE-2021-45046. Deze kwetsbaarheid vereist wat specifieke omstandigheden, de kans dat een aanval succesvol kan worden uitgevoerd is lager. Voor de 'severity' van een kwetsbaarheid wordt een zgn. 'CVSS' score gebruikt, ter vergelijking: De CVE-2021-44228 'Log4Shell' kwetsbaarheid krijgt een 10.0 (hoger gaat het niet), CVE-2021-45046 krijgt een rating van 9.0.

Er is ook een nieuwe Log4j 2.x kwetsbaarheid ontdekt, CVE-2021-45105 met een CVSS score van 7.5. Door deze kwetsbaarheid is het mogelijk om, onder bepaalde omstandigheden, de applicatie te laten crashen. Niet elke applicatie is hier vatbaar voor, het hangt af van wat er precies wordt gelogd. Dat is natuurlijk vervelend, maar staat qua risico niet in verhouding tot kwetsbaarheden waarbij code kan worden uitgevoerd. Deze kwetsbaarheid is opgelost in log4j versie 2.17 (dus voor Java 8 of hoger). Voor Java 7 is nog geen update, maar het ligt in de lijn der verwachting dat hier t.z.t. een 2.12.3 versie voor wordt uitgebracht.

7. Workarounds

We gaan hier kort in op workarounds voor de eerder genoemde kwetsbaarheden.

7.1 Log4j 1.x

Kwetsbaarheid	Omschrijving workaround
CVE-2019-17571	Oude, maar wel serieuze kwetsbaarheid. Heeft geen raakvlakken met CVE-2021-44228 'Log4Shell'. Geen workaround bekend.
CVE-2021-4104	Heeft wat raakvlakken met CVE-2021-44228. Zorg ervoor dat er geen JMSAppender is geconfigureerd.

7.2 Log4j 2.x

Kwetsbaarheid	Omschrijving workaround
CVE-2021-44228	• Een genoemde workaround is om log4jLog4j2.formatMsgNoLookups op true te zetten. Het risico wordt hiermee lager, maar dit mitigeert echter niet elke 'attack vector'. Deze workaround werkt niet voor een Log4j versie lager dan 2.10 . • Verwijder de JndiLookup class. Deze workaround biedt wel complete mitigatie.
CVE-2021-45046	Heeft raakvlakken met CVE-2021-44228. Verwijder de JndiLookup class. Deze workaround biedt complete mitigatie.
CVE-2021-45105	Nog geen workaround voor bekend. Update naar Log4j 2.17 (Java 8). Voor Log4j 2.12 (Java 7) is nog geen update.

8. Mitigatie met een Web Application Firewall (WAF) of Intrusion Prevention Systems (IPS)

Veel omgevingen gebruiken een Web Application Firewall (WAF) of Intrusion Prevension System (IPS). Deze dienst staat in-line in het netwerkverkeer en kan aanvallen detecteren en blokkeren. De WAF/IPS oplossingen gebruikt door Solvinity hebben signatures vrijgegeven voor deze aanval, deze zijn geïnstalleerd en worden automatisch up-to-date gehouden. Echter, hoewel een WAF of IPS security toevoegt, is het is goed om hier niet alleen op te vertrouwen. Voor deze aanval worden aan de lopende band 'evasion' technieken ontwikkeld om ervoor te zorgen dat de WAF of IPS de aanval niet detecteert. Wanneer er nieuwe signatures beschikbaar komen die dit toch kunnen detecteren, worden deze automatisch geïnstalleerd, maar het blijft een kat- en muisspel. Daarnaast vindt deze aanval meestal plaats over het versleutelde HTTPS, er moeten dus maatregelen zijn genomen om ervoor te zorgen dat de WAF of IPS dit toch kan inspecteren.

9. Hoe nu verder ?

Deze kwetsbaarheid is nog vrij nieuw. Er komen mogelijk nog meer manieren om hier misbruik van te maken. Advies van Solvinity is om waar Log4j wordt gebruikt zo snel mogelijk de workaround te implementeren (verwijder de JndiLookup class) of te patchen. In beide gevallen moet daarna de applicatie herstart worden.

Bij gebrek aan resources is het qua volgorde van werkzaamheden in de meeste gevallen verstandig eerst de update of workaround door te voeren op applicaties met een Log4j versie onder 2.15. Daarna kunnen werkzaamheden worden uitgevoerd om Log4j te updaten naar de laatste versie.

Solvinity bouwt geen Java applicaties, maar we maken er wel gebruik van. Daar waar wij zelf Java applicaties gebruiken, voeren we een update uit of passen we de workaround toe. Dit zijn onze eigen interne applicaties, maar ook onderliggende applicaties van onze diensten.

Voor applicaties van klanten geldt dat Solvinity hier niet primair verantwoordelijk voor is, maar we ondersteunen onze klanten zo goed als mogelijk bij het detecteren van kwetsbare log4j versies en het updaten ervan. Daar waar Solvinity toegang en inzicht heeft controleren we, automatisch en/of handmatig, op kwetsbare Log4j versies. In overleg met de klant passen we een workaround toe of doen we een update. Dit ontslaat klanten niet van hun verantwoordelijkheid om zelf actie te ondernemen. Log4j kan goed 'verstopt' zitten, in alle gevallen moet de bouwer van de applicatie worden geraadpleegd om er zeker van te zijn dat er geen kwetsbare Log4j wordt gebruikt. Sommige Java applicaties komen bijvoorbeeld als één applicatiebestand, waar Log4j in verpakt zit. Dit is een stuk lastiger te detecteren. Het kan ook zijn dat de Log4j library is aangepast door de applicatieontwikkelaar waardoor de versie niet goed uitgelezen kan worden.

Al met al is dit een kwetsbaarheid die echt serieus genomen moet worden, de verwachting is dat we hier nog lang over horen en lezen.

10. Externe bronnen

- https://unit42.paloaltonetworks.com/apache-log4j-vulnerability-cve-2021-44228/
- https://blog.cloudflare.com/inside-the-log4j2-vulnerability-cve-2021-44228/
- https://logging.apache.org/log4j/2.x/security.html
- https://logging.apache.org/log4j/2.x/changelog.html
- https://github.com/NCSC-NL/log4shell
- https://www.ncsc.nl/actueel/advisory?id=NCSC-2021-1052