Säkerhetsnätet som aldrig sover

En fallstudie i Vulnerability Scanning.

Examensarbete 30 YH-poäng

Internet of Things Developer

Göteborg 2023

Sebastian Andersson

Datum: 2023-05-22
Handledare: Andreas Nilsson Ström

Förord

Jag vill uttrycka min ambition med denna rapport och samtidigt framföra tacksamhet till de personer och organisationer som bidragit till mitt intresse för IT-säkerhet. Detta förord är en inledning till det.

Under våren 2023 genomförde jag min LIA-period (Lärande i arbete) på CShift i Göteborg. Där väcktes mitt intresse för IT-säkerhet på ett sätt som jag inte hade kunnat föreställa mig. Genom deras stora engagemang öppnades dörren till en värld av tekniska utmaningar och möjligheter. Detta är jag väldigt tacksam för.

Ambitionen med denna rapport har varit att använda den skarpa kniven istället för smörkniven. Det innebär att jag strävat efter att vara så konkret och precis som möjligt i mitt arbete. Genom att fokusera på detaljer och tydlig analys har jag hoppats kunna leverera en rapport som inte bara är informativ, utan även användbar och relevant inom IT-säkerhetsområdet.

Jag vill även ta tillfället i akt att rikta ett tack till KYH och mina fantastiska klasskompisar som hela tiden stöttat, tröstat och utmanat. Det är inte helt lätt att börja programmera!

Särskilt vill jag rikta mitt djupaste tack till medarbetarna på CShift för deras enorma engagemang och stöd. Carl Retzner och Jimmie Ekvall, som varit mina närmaste handledare, har varit ovärderliga i min tekniska utveckling. Deras kompetens och tålmodighet har varit en ständig inspiration för mig. Jag vill också tacka Christian Olivefors och Oscar Jacobson för deras ovärderliga stöd och vägledning som har hjälpt mig att växa både som yrkesperson och människa.

Ett stort tack till er alla för att ni trott på mig, utmanat mig och stöttat mig under hela resans gång. Utan ert engagemang och stöd hade jag inte kunnat nå så långt som jag nu har gjort.

Slutligen riktar jag ett tack till alla som bidragit till min rapport och till alla framtida läsare som jag hoppas ska inspireras till en karriär inom IT-säkerhet. Det är otroligt roligt, och du behövs!

Tack!

Göteborg
2023-05-22
Sebastian Andersson

Sammanfattning

Denna rapport fokuserar på området Vulnerability Management och ämnar att undersöka och utvärdera en specifik del inom detta område; Vulnerability Scanning. Med ramverket Vimp utförs analyser av skanningverktygen Grype, Snyk och Trivy i syfte att analysera skillnaderna i verktygens klassificering och filtrering av sårbarheter. Om verktygen visar olika information, vad betyder det för organisationers förmåga att bedöma säkerhetshot?

För att ge en gemensam utgångspunkt för bedömningen av sårbarheter används FIRST.org och deras CVSS-poängsystem. Tidigare säkerhetsincidenter tas även upp för att belysa vikten av ett proaktivt och levande säkerhetsarbete för att förebygga sårbarheter.

Rapporten syftar inte till att ge en djupgående förståelse av Vulnerability Management, IT-säkerhet, tidigare säkerhetsincidenter eller OSS. Däremot syftar den till att på ett konkret sätt jämföra en och samma kodbas mot tre olika skanningsverktyg för att utvärdera och diskutera skillnaderna i resultatet. Förhoppningen är att detta ska ge en fördjupad förståelse i komplexiteten och relevansen av att “shift left” för att integrerar IT-säkerhet i ett tidigare skede i utvecklingsprocessen.

Abstract

This report focuses on the area of Vulnerability Management and aims to investigate and evaluate the specific field of Vulnerability Scanning. Within the framework of Vimp analyses are conducted using the scanning tools Grype, Snyk, and Trivy in order to examine the differences in their classification and filtering of vulnerabilities. If the tools provide different information what does it mean for organizations' ability to assess security threats?

To establish a common baseline for vulnerability assessment, the report utilizes the FIRST.org and their Common Vulnerability Scoring System (CVSS) ratings. Previous security incidents are also addressed to highlight the importance of proactive and ongoing security measures to prevent vulnerabilities.

It is important to note that the report does not aim to provide an in-depth understanding of Vulnerability Management, cybersecurity, previous security incidents, or Open Source Software (OSS). However, its specific goal is to compare the same codebase against three different scanning tools to evaluate and discuss the differences in results. The aim is that this analysis will provide a deeper understanding of the complexity and relevance of “shifting left” to integrate cybersecurity at an earlier stage in the development process.

Innehållsförteckning

1. Inledning **6**
1.2 Bakgrund 6
1.2.1 Vad är Vulnerability Management? 6
1.2.2 Varför är det viktigt? 7
1.2.3 Några varnande exempel från förr 8
2. Syfte **9**
2.1 Problemformulering 9
2.2 Avgränsningar 10
2.3 Metod 10
3. Teori **10**
3.1 Snyk, Grype och Trivy 10
3.2 Vimp 11
3.3 FIRST.org och CVSS 11
3.4 React 11
4. Beräkningar och resultat **12**
4.1 Skanning 13
4.2 CVSS High eller Critical 14
4.3 CVSS Critical med fördjupning 15
5. Slutsatser **17**
Källförteckning **19**
Bilaga 1 **22**
Bilaga 2 **25**
Bilaga 3 **27**
Bilaga 4 **30**

1. Inledning

Vulnerability Management har blivit en allt viktigare aspekt inom IT-säkerhet och det syftar till att identifiera och hantera sårbarheter i tekniska system. Även om siffrorna varier smått mellan olika källor beräknas ca 90 % av all världens kodbas innehålla Open Source Software (OSS) och det ser ut att fortsätta öka (Kocher, 2023 - Ahmed, 2022 - Pump, 2022 - Octoverse, 2022). Medan OSS erbjuder många fördelar, som återbruk och kostnadsbesparingar, medför det även säkerhetsrisker. Bristen på insyn i OSS-projekt kan leda till problem när det är oklart vem som står ansvarig för projektet, vilken kompetens de har eller hur ofta koden underhålls.

För att belysa relevansen av Vulnerability Management kommer vi att titta på några välkända exempel. Log4J (Gallo, 2022) och Heartbleed (Gajawada, 2016) skapade stora problem och visade de allvarliga konsekvenser som säkerhetsbrister kan ha. Dessa händelser har varit en påminnelse om behovet av robusta och effektiva sätt att upptäcka och hantera sårbarheter för att skydda våra system och data.

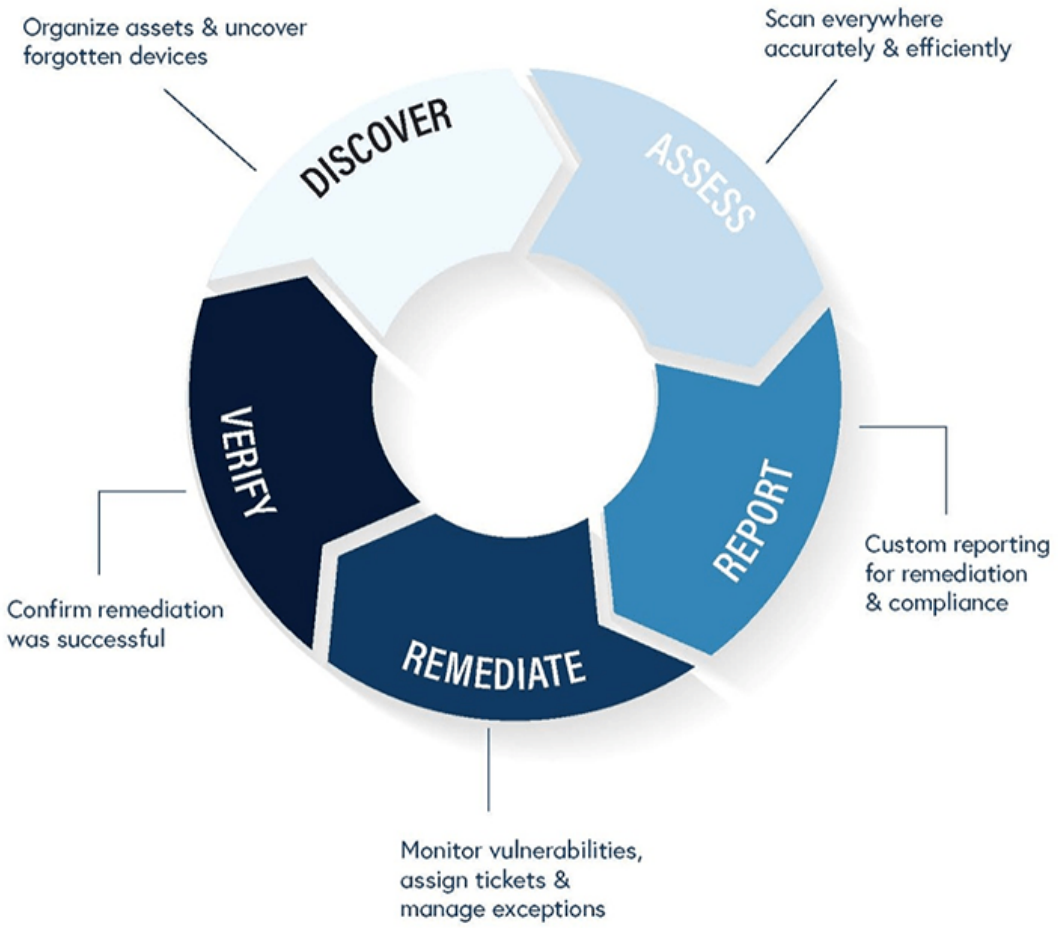
Mot bakgrund av detta har jag som mål att utforska och analysera sårbarheter i ett nystartat React-projekt genom att använda 3 olika Vulnerability Scanners. Genom att jämföra verktygen på samma kod avser jag inte enbart att undersöka hur ett sårbarhetstest genomförs, men också identifiera eventuella skillnader mellan dem.

Det är dags att “shift left” (Mcbride, 2022) - flytta fokus och insatser för Vulnerability Management till en tidigare fas i utvecklingsprocessen. Genom att integrera säkerhet på ett tidigt stadium och inkludera sårbarhetstester i utvecklingscykeln kan vi minimera riskerna och stärka våra system mot potentiella hot.

1.2 Bakgrund

1.2.1 Vad är Vulnerability Management?

Vulnerability Management (VM) är en kontinuerlig och proaktiv process vars syfte är att hålla tekniska system säkra från cyberattacker och dataintrång. Det främsta målet är att minimera riskexponering genom att hantera sårbarheter. Det finns flera olika sätt att minimera riskexponeringen för ett system, däri inkluderat att identifiera, bedöma och förebygga. Det finns flera olika representationer av Vulnerability Management, och figur 1 är en bra modell som är lånad (TEAM ASCEND, 2019):



Figur1. Modell för Vulnerability Management.

Vulnerability Management innefattar flera viktiga komponenter, såsom inventering av tillgångar, sårbarhetsskanningar, uppdateringshantering, konfigurationshantering, hantering av säkerhetsincidenter, penetrationstestning och mer. I denna rapport kommer vi att fokusera främst på sårbarhetsskanningar som ett verktyg inom Vulnerability Management (Microsoft, 2023).

1.2.2 Varför är det viktigt?

En av de främsta anledningarna till att Vulnerability Management är viktigt är att det ger organisationer möjlighet att utvärdera säkerhetsrisker och därmed prioritera resurser för att åtgärda dem på rätt plats och i rätt tid. Välkända säkerhetssårbarheter utnyttjas ofta av illvilliga aktörer, vilket kan leda till allvarliga säkerhetsincidenter vilket kan kosta pengar och skada relationer. Dessutom riskerar organisationer att bryta både kontraktsrelaterade och juridiska krav om de försummar sitt säkerhetsarbete (National Cyber Security Centre, 2023).

Därtill behöver Open Source Software (OSS) belysas inom ramen för Vulnerability Management. En studie utförd av Synopsys (Kocher, 2023) visar att nästan all programvara innehåller

någon form av öppen källkod, och att 48% av koden hade allvarliga säkerhetsbrister. Forskare fann vidare att 84% av kodexempel hade minst en känd sårbarhet, inklusive “critical”[1]. Studien analyserade 1481 källkoder och fann att 73% av koden var öppen källkod. Olika branscher, som flyg- och rymdindustrin, fordonsindustrin, transport- och logistikbranschen, visade en ökning av användningen av öppen källkod och högrisk-sårbarheter under de senaste fem åren. Dåligt underhållna eller säkerhetsmässigt “svaga” (föråldrade) komponenter hittades i 91% av de utvärderade källkoderna, vilket tyder på ett behov av bättre underhåll och uppdateringar. Även om olika studier kommer fram till smått varierande resultat i procentsiffror, så är de fortfarande lika i sina uppskattningar (Ahmed, 2022 - Pump, 2022 - Octovese, 2022).

Det bör även nämnas att några av flera problem med OSS inkluderar potentiell brist på säkerhetskompetens i utveckling av kod, licenshantering vilket ökar risken för användandet av kod enligt ett av licensen felaktigt sätt och det faktum att koden finns publikt för många att få tillgång till. Det senare öppnar upp för illvilliga aktörer att aktivt söka efter sårbarheter (Mcbride, 2022).

1.2.3 Några varnande exempel från förr

För att belysa konsekvenserna av bristfällig IT-säkerhet ska vi nu titta på några exempel på säkerhetsincidenter som har haft allvarliga följder. Vi börjar med fallet Heartbleed, en säkerhetsbugg som upptäcktes första gången 2014 och som grundade sig i en sårbarhet i OpenSSL. OpenSSL är en Open Source Software som används för att kryptera kommunikation och används av många olika program och aktörer.

Några välkända företag och plattformar som drabbades av Heartbleed-incidenten var Google, Yahoo, Amazon Web Services (AWS), Instagram och Netflix. Genom denna sårbarhet kunde angripare utnyttja ett fel i Heartbleeds meddelandehantering och få tillgång till känslig information, såsom krypteringsnycklar, lösenord och användarnamn. Detta var möjligt på grund av brister i begränsningar för "requests", vilket gjorde att angripare kunde få Heartbleed att skicka tillbaka stora mängder data som egentligen inte skulle ha avslöjats (Gajawada, 2016).

“During a TLS encrypted handshake, two machines send each other Heartbeat messages. According to RFC 6520, a Heartbeat response needs to contain the exact copy of the payload from the Heartbeat request. When a Heartbeat request message is received, the machine writes the payload contents to its memory and copies the contents back in response. The length field is meant to be the length of the payload. OpenSSL allocates memory for the response based on length and then copies the payload over into the response using memcpy().”(Gajawada, 2016)

[1] CVSS är en väletablerad “ranking” av allvarlighetsgrad för sårbarheter i kod.

Idag är detta inte ett problem för OpenSSL och buggen, när den väl upptäcktes, kunde relativt enkelt åtgärdas. Trots detta läckte känslig information som hade potential att göra enorm skada.

I december 2021 upptäcktes en annan sårbarhet som kom att benämnas som Log4J. Log4J är ett loggningsverktyg utvecklat av Apache som används för att logga kommunikation inom system. Sårbarheten, som upptäcktes av en tillfällighet, gör att angripare kan injicera skadlig kod i loggarna. På grund av Log4Js förmåga att kommunicera inom systemet kunde angripare exekvera kod i andra system som Log4J hade kontakt med. Sårbarheten upptäcktes först genom spelet Minecraft när användare insåg att spelchatten loggades med Log4J och kunde användas för att exekvera kod.

> *You might be wondering, "What's so special about the crafted payload?" The payload is ultimately what exploits the Log4j vulnerability. It does so by using JNDI, Java Naming and Directory Interface, a feature that enables a user to fetch and load Java objects from a server. Although this is a secure functionality, the Log4j flaw allows an attacker to input their own JNDI lookups, where they then direct the server to their fake LDAP server. From here, the attacker now has control of the remote system and can execute malware, exfiltrate sensitive information like passwords, and more. (Gallo, 2022)*

Konsekvenserna av Log4J är fortfarande kännbara än idag och sårbarheten påverkade en väldigt stor mängd system och personer vilket inkluderade federala myndigheter i USA (ZDNet (2023), Microsoft Azure (GeekforGeeks, 2023) och flertalet myndigheter inom EU (ENISA, 2023). Detta var några av flera exempel som belyser de stora problem som Log4J incidenten medförde.

2. Syfte

Syftet med denna rapport är att belysa området för sårbarhetshantering, men också att på ett konkret sätt utvärdera en aspekt av det - nämligen sårbarhetsskanning. Genom att använda en och samma kodbas för skanning med tre separata verktyg kan vi jämföra skillnaderna mellan dem. Vi kan också använda FIRST.org CVSS-poäng som en gemensam utgångspunkt för att bedöma allvarligheten hos incidenter. Det är dock rimligt att förvänta sig att det kommer att finnas skillnader i vad som inkluderas i en skanningstjänst, och vi kommer att diskutera orsakerna till detta.

2.1 Problemformulering

Hur skiljer sig de tre verktygen för sårbarhetsskanning; Grype, Trivy och Snyk i deras förmåga att fånga upp sårbarheter och vad betyder det för vår förmåga att utvärdera säkerhetshot?

2.2 Avgränsningar

Denna rapport ämnar inte att ge en heltäckande redogörelse för sårbarhetshantering - det skulle vara en egen rapport i sig. Istället fokuserar vi på sårbarhetsskanning och användningen av tre vanligt förekommande verktyg.

Även om Open Source Software (OSS) har stor relevans inom sårbarhetshantering och IT-säkerhet, är det inte den enda formen av kod som används eller återfinns vid säkerhetsincidenter. Ett projekt som specifikt behandlar OSS och IT-säkerhet skulle vara intressant, men det ligger utanför ramen för denna rapport, även om det nämns.

Jag har valt att lyfta fram några säkerhetsincidenter i bakgrundsavsnittet. Syftet med detta är att belysa vikten av ett proaktivt och levande säkerhetsarbete för att förebygga sårbarheter. Därför ger vi en översiktlig beskrivning av händelserna, eftersom rapportens huvudsakliga syfte aldrig har varit att ingående analysera dem, utan bara att visa dem som varnande exempel på vad som kan hända om man försummar sitt säkerhetsarbete.

2.3 Metod

Jag utför en jämförande fallstudie genom att applicera 3 välkända Vulnerability Scanning-verktyg på en React-applikation för att utvärdera deras prestation. Eftersom jämförelsepunkten (React) är densamma för samtliga verktyg, borde jag kunna identifiera skillnader mellan hur verktygen fungerar.

3. Teori

Ramverken för min analys är tre skanningsverktyg som kommer att exekveras genom ett verktyg som heter Vimp. Jämförelsen mellan verktygen blir som mest relevant med vetskapen om att samtliga verktyg utgår från samma poängsättning av allvarlighetsgrad för sårbarheter, nämligen CVSS. I detta kapitel kommer jag att redogöra i detalj för verktygens användningsområden och kapacitet, hur de implementeras i verktyget Vimp och hur jämförelser kan göras med utgångspunkt i CVSS.

3.1 Snyk, Grype och Trivy

Verktygen erbjuder flera tjänster för att identifiera och åtgärda sårbarheter i källkod. Samtliga erbjuder integrering i CI/CD flöden och stödjer flera olika programmeringsspråk. Systemen kan dels sårbarhetskanna aktuell kod, men även överblicka “dependencies”[2] i syfte att uppnå en djupare analys. Därtill erbjuder tjänsterna aktiv skanning för projekt tillsammans med rekommendationer om åtgärder för att hantera säkerhetsbrister.

[2] Egen kommentar: i detta fall kodpaket som är integrerad i kodpaket. Flerlagersskanning möjliggör en djupare analys och riskvärdering.

Det finns flera olika sätt att ta del av filtrerad information efter skanning, som t.ex. en JSON-rapport eller CSV-format. Ur en riskbedömningsaspekt grundar sig samtliga tre på FIRST.org's CVSS poängskala (Snyk, 2023 - Özkan, 2022 - Anchorage, 2023 - Aqua Security, 2023). Detta underlättar vår jämförelse av verktygens output.

3.2 Vimp

Vimp är ett verktyg som syftar till att underlätta sårbarhetsskanningar från flera olika källor. Specifikt fokuserar Vimp på integrationen med tre utvalda verktyg: Snyk, Grype och Trivy. Genom att sammanställa informationen från dessa skanningar erbjuder Vimp möjligheten att överskådligt jämföra och analysera skillnaderna i resultat mellan de olika skanningsverktygen. Rapportresultaten kan erhållas i olika format, men i denna rapport har jag valt att använda JSON-filer då de erbjuder en lättläst och tydlig struktur (McMarny, 2023).

3.3 FIRST.org och CVSS

CVSS (Common Vulnerability Scoring System) är en standard för att beräkna allvarlighetsgrad för sårbarheter i kod. Projektet drivs av FIRST.org men implementeras av alla stora Vulnerability Management system (Common Vulnerability Scoring System, 2023). Poängsystemet är en skala från 0-10 med tillhörande skriftlig kategori. Se bild 2 för förtydligande:

Severity	Base Score Range
None	0.0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

Tabell 1. CVSS sammanfattning

Den absolut största majoriteten av alla kända sårbarheter finns lagrade i The National Vulnerability Database (NVD). På detta sätt kan man uppnå en standardiserad och "objektiv" klassificering och överblick av sårbarheter (Common Vulnerability Scoring System, 2023).

3.4 React

Jag kommer att använda mig av en React-app som grund för min sårbarhetsskanning. React är ett javascript-bibliotek utvecklat av Meta (dåvarande Facebook) och möjliggör smidig webbutveckling. Det är väldigt välanvänt. Vid skapandet av React-appen[3] kan vi räkna antal paket importerad kod som följer med i instansieringen av appen genom att skriva `ls node_modules | wc -l` i vår root-folder. Hela

[3] Grundversionen skapad 2023-05-30, 17:42

825 paket importerad kod finns från start, och syftet med denna rapport är att ta reda på hur många av dem som är allvarliga sårbarheter, och hur skanningverktygen skiljer sig åt i sin bedömning.

4. Beräkningar och resultat

Studien inleds genom installation av koden som används för sårbarhetsskanning, vilket är React. React är ett ramverk för webbutveckling. Vid installationen av appen erhålls vissa grundinställningar för hantering av en inledande landningssida och diverse konfigurationer.

Först installeras Node (Node, 2023). Genom att navigera till terminalen används WSL (Windows Subsystem for Linux) för att använda verktygen (WSL, 2023).

En ny mapp med namnet "examensarbete" skapas för projektet genom att köra kommandot:

```
mkdir examensarbete
```

Därefter följer installationen av React-appen med namnet react_examen:

```
cd examensarbete
```

följt av:

```
npx create-react-app react_examen
```

fortsätt att navigera till react_examen:

```
cd react_examen
```

Samtliga dependencies (det vill säga importerad kod till React) installeras genom att skriva:

```
npm install
```

Nu är React-appen installerad. Installation av Snyk genomförs med kommandot:

```
npm install -g snyk
```

Skapa ett konto på Snyks hemsida för att därefter logga in på kontot genom att skriva:

```
snyk auth
```

Installera Grype genom kommandot:

```
curl -sSfL
https://raw.githubusercontent.com/anchore/grype/main/install.sh | sh -s
-- -b /usr/local/bin
```

För att använda Grype behövs inget användarkonto. Det sista skanningsverktyget att installera är Trivy genom kommandot:

```
wget
https://github.com/aquasecurity/trivy/releases/download/v0.18.3/trivy_0.
18.3_Linux-64bit.deb
```

följt av:

```
sudo dpkg -i trivy_0.18.3_Linux-64bit.deb
```

Trivy behöver inte heller ett användarkonto för att fungera.

4.1 Skanning

En fullständig rapport i JSON-format som skannar React-projektet kan skrivas ut (Se bilaga 1, 2 och 3). För att åstadkomma detta, körs följande kommandon för att använda alla skanningverktyg:

```
grype --add-cpes-if-none -s AllLayers -o json --file report.json $image
snyk container test --app-vulns --json-file-output=report.json $image
trivy image --format json --output report.json $image
```

Resultatet från vår lokala fil importeras genom en Docker-image och körs mot Vimps egna databas. Det resulterande resultatet sparas i en SqLite-databas. I denna skanning används alla verktyg för att totalt hitta 56 unika sårbarheter på CVSS-skalan LOW-CRITICAL. Informationen erhålls genom att köra en fråga mot vår SqLite-databas, där vi räknar antalet unika CVE-id som är sparade. Varje sårbarhet har ett unikt CVE värde, vilket underlättar vid skanning (se bilaga 4).

```
sqlite> SELECT COUNT(DISTINCT exposure) AS count
FROM vul;
56
```

Resultatet kan brytas ner för varje enskild skanner genom att använda frågor mot SqLite-databasen. Se antalet unika sårbarheter som verktyget Snyk har identifierat i figur 2, vilket är 48 stycken:

```
sqlite> SELECT COUNT(DISTINCT exposure) AS count
FROM vul
WHERE source IN ('snyk');
48
```
Figur 2 Snyk generell

Om en körning genomförs mot databasen och vi byter ut "Snyk" mot "Grype" på följande rad `WHERE source IN ('snyk');` får vi ett resultat på 54 unika sårbarheter. Detta visar på en skillnad jämfört med resultatet från Snyk.

```
sqlite> SELECT COUNT(DISTINCT exposure) AS count
FROM vul
WHERE source IN ('grype');
54
```
Figur 3 Grype generell

I figur 4 ser vi resultatet från Trivy:

```
sqlite> SELECT COUNT(DISTINCT exposure) AS count
FROM vul
WHERE source IN ('trivy');
52
```
Figur 4 Trivy generell

Beräkningarna visar att de olika verktygen inkluderar olika antal sårbarheter i sina skanningar, vilket i sig inte behöver vara överraskande. Eftersom skanningarna hittills har fokuserat på sårbarheter inom CVSS-intervallet LOW-CRITICAL är det inte särskilt problematiskt om variationen ligger i sårbarheter på den lägre delen av skalan (t.ex. LOW-MEDIUM). Nu kommer jag att undersöka om det finns skillnader mellan verktygen när det gäller sårbarheter högre upp på CVSS-skalan.

4.2 CVSS High eller Critical

Vid användandet av Snyk och sökning efter unika sårbarheter som använder källan Snyk, men som även har ett severity-värde på “HIGH” eller “CRITICAL” visas resultatet i figur 5.

```
sqlite> SELECT COUNT(*) AS total_rows
FROM (
  SELECT DISTINCT exposure
  FROM vul
  WHERE source = 'snyk'
    AND severity IN ('high', 'critical')
) AS subquery;
```

```
24
```

Figur 5 Snyk high eller ciritical

Resultatet för Grype och skanning har inkluderat fler unika sårbarheter och kan observeras i figur 6:

```
sqlite> SELECT COUNT(*) AS total_rows
FROM (
  SELECT DISTINCT exposure
  FROM vul
  WHERE source = 'grype'
    AND severity IN ('high', 'critical')
) AS subquery;
29
```

Figur 6 Grype high eller critical

Och sedan samma sak för Trivy som hittar 10 unika sårbarheter:

```
sqlite> SELECT COUNT(*) AS total_rows
FROM (
  SELECT DISTINCT exposure
  FROM vul
  WHERE source = 'trivy'
    AND severity IN ('high', 'critical')
) AS subquery;
10
```

Figur 7 Trivy high eller critical

4.3 CVSS Critical med fördjupning

I denna mer detaljerade beräkning undersöks antalet unika sårbarheter som varje skanningsverktyg presenterar med en allvarlighetsgrad ("severity") av "CRITICAL". Tillsammans med detta efterfrågas också CVE-id för att analysera vilka sårbarheter som inkluderas eller exkluderas av olika skanningar och verktyg.

Vid en närmare observation av Snyk identifieras 3 “CRITICAL” sårbarheter:

```
sqlite> SELECT exposure, COUNT(*) AS count
FROM vul
WHERE source = 'snyk' AND severity = 'critical'
GROUP BY exposure;
CVE-2005-2541
CVE-2019-1010022
CVE-2019-8457
```

Figur 8 Snyk critical

Därefter Grype som hittar 4 stycken:

```
sqlite> SELECT exposure, COUNT(*) AS count
FROM vul
WHERE source = 'grype' AND severity = 'critical'
GROUP BY exposure;
CVE-2019-1010022
CVE-2019-8457
CVE-2022-0543
CVE-2022-3734
```

Figur 9 Grype critical

Och Trivy som bara lyckades hitta 1:

```
sqlite> SELECT exposure, COUNT(*) AS count
FROM vul
WHERE source = 'trivy' AND severity = 'critical'
GROUP BY exposure;
CVE-2019-8457
```

Figur 10 Trivy critical

Sammanfattning av resultatet kan ses i tabell 2:

CVE-id	***SNYK***	***GRYPE***	***TRIVY***	***Omtvistad***
CVE-2005-2541	Ja	Nej	Nej	X
CVE-2019-1010022	Ja	Ja	Nej	Xgma
CVE-2019-8457	Ja	Ja	Ja	X
CVE-2022-0543	Nej	Ja	Nej	
CVE-2022-3734	Nej	Ja	Nej	X

Tabell 2. Jämförelse av kritiska sårbarheter.

Nu presenteras en närmare titt på samtliga sårbarheter med information från NIST.

CVE-2005-2541 är en sårbarhet i TAR 1.15.1 som brister i varningar när man extraherar setuid eller setgid-filer. Detta kan ge angripare förhöjda privilegier i systemet. Det är dock värt att notera att sårbarheten för närvarande utvärderas på nytt vid skrivandet av denna text och kan få en annan klassificering senare (CVE-2005-2541, 2023).

CVE-2019-1010022 är associerad med GNU Libc och är en sårbarhet som kan ge angriparen tillgång till systemet genom en stack buffer overflow. Det möjliggör potentiell manipulation av en buffer/array/minnesallokering, vilket öppnar möjligheter att få tillgång till eller ändra information som

ligger utanför det bestämda minnesspannet. Det skulle kunna leda till liknande problem som det tidigare nämnda Log4J. Det är dock viktigt att påpeka att även denna sårbarhet för närvarande utvärderas på nytt, så bedömningen kan ändras (CVE-2019-1010022, 2023).

CVE-2022-0543 är en sårbarhet som påverkar Redis-databaser. Den klassificeras som ett Lua sandbox escape vulnerability, vilket innebär att en angripare skulle kunna bryta sig ur en säker miljö för att kunna exekvera kod gentemot delar av systemet som normalt inte ska vara åtkomliga (CVE-2022-0543, 2023).

CVE-2022-3734 är också associerad med Redis, där manipulering av filen "C:/Program Files/Redis/dbghelp.dll" kan skapa en okontrollerad sökväg och ge angripare möjlighet att oavsiktligt exekvera kod. Det bör dock noteras att även denna sårbarhet för närvarande utvärderas på nytt, så bedömningen kan ändras (CVE-2022-37-34, 2023).

Den sista identifierade sårbarheten är CVE-2019-8457, en out-of-bounds sårbarhet i rtreenode()-funktionen. I detta fall innebär en out-of-bounds sårbarhet felaktig eller oväntad inläsning från databasen, vilket kan skapa problem då angripare kan få tillgång till skyddad information. Det bör dock nämnas att även denna sårbarhet för närvarande utvärderas igen och klassificeringen kan ändras (CVE-2019-8457).

5. Slutsatser

I detta avsnitt har undersökning av sårbarhetshantering och skannerverktyg utforskats och slutsatser har presenterats baserade på undersökningen. Genom historiska exempel har betydelsen av effektiv hantering av sårbarheter i IT-system och applikationer belysts.

För att visa förmågan hos olika sårbarhetsskanningsverktyg att identifiera och hantera sårbarheter valdes en React-applikation som kodexempel. Genom att köra denna applikation med tre olika skannerverktyg - Grype, Trivy och Snyk - analyserades och jämfördes deras resultat.

Skillnader noterades i verktygens förmåga att upptäcka och rapportera unika sårbarheter. Verktygen identifierade olika antal sårbarheter, vilket var tydligt vid olika steg på CVSS-skalan. Detta är intressant och en fördjupad studie skulle kunna undersöka varför resultaten varierade.

Fokus lades särskilt på sårbarheter med allvarlighetsgraden "CRITICAL". Genom att sammanställa en tabell som tydligt visar vilka sårbarheter som identifierades av varje skannerverktyg med respektive CVE-id observerades intressanta mönster och skillnader i resultaten.

För att få en djupare förståelse för varje sårbarhet söktes deras CVE-id hos National Institute of Standards and Technology (NIST). Detta gav tillgång till mer detaljerad information om sårbarheterna. Det är dock viktigt att notera att vid skrivandet av denna rapport är fyra av de fem identifierade sårbarheterna fortfarande under ny utvärdering hos NIST. Detta kan förklara variationen i de resultat som presenteras av skannerverktygen. Mot bakgrund av detta blir det ännu viktigare att

vidta åtgärder för att hantera potentiella sårbarheter enligt principen "better safe than sorry", även om deras exakta klassificering ännu inte är fastställd.

En intressant observation är att den kritiska sårbarheten med CVE-id CVE-2022-0543 endast identifierades av ett av skannerverktygen, nämligen Grype. Detta resultat är anmärkningsvärt och väcker frågor om skannerverktygens olika förmågor att upptäcka specifika sårbarheter. Det betonar också vikten av att använda en varierad uppsättning verktyg och metoder för att få en heltäckande bild av hantering av sårbarheter.

Generellt sett har Grype presterat väl i denna studie, särskilt när det gäller sårbarheter med hög allvarlighetsgrad. Detta resultat betonar vikten av att använda en varierad uppsättning verktyg och vara medveten om deras olika styrkor och svagheter vid hantering av sårbarheter. För mer information, se tabell 3 för att se resultatet.

CVSS	***Serverity***	***Grype***	***Snyk***	***Trivy***
0-10	LOW-CRITICAL	54	48	52
7-10	HIGH-CRITICAL	29	24	10
9-10	CRITICAL	4	3	1

Tabell 3. Sammanfattning sårbarheter för samtliga verktyg.

Det är viktigt att notera att trots de olika resultaten som verktygen presterar, är det svårt att dra stora slutsatser om orsakerna bakom deras inkludering eller exkludering av vissa resultat. Motivationen bakom deras val ligger utanför ramen för denna rapport. Det framgår dock tydligt att Grype verkar ge en bredare överblick och presterar bäst i detta test, vilket gör det till ett föredraget verktyg.

Denna studie har gett oss värdefulla insikter och kunskaper om sårbarhetshantering och skannerverktyg. Resultaten betonar vikten av att bedöma och använda flera verktyg samt att förstå de unika egenskaperna hos varje verktyg vid hantering av sårbarheter.

Avslutningsvis är det viktigt att betona vikten av att "shift left" för att integrera ett konkret och systematiskt säkerhetstänkande tidigt i utvecklingsprocessen. Det är viktigt att komma ihåg att denna skanning baserades på en grundkonfiguration av en React-applikation. Om vi kan identifiera kritiska sårbarheter i en grundkonfiguration hos en välrenommerad tjänsteleverantör, är det inte svårt att föreställa sig att stora IT-projekt kan ha en betydande mängd potentiella sårbarheter. Med tanke på dagens digitala infrastruktur är det absolut nödvändigt att utvecklare har en gedigen säkerhetsmedvetenhet genom hela utvecklingsprocessen.

Källförteckning

- **Ahmed, A. (2022).** Open-Source Software are everywhere, confirms a new report. Hämtad från Digital Information World: https://www.digitalinformationworld.com/2022/11/open-source-software-are-everywhere.html (Hämtad 2023-05-23 kl. 09:15)
- **Anchore. (2023).** Grype Repository. Hämtad från GitHub: https://github.com/anchore/grype (Hämtad 2023-05-25 kl. 14:30)
- **Anil Gajawada. (2016).** Heartbleed bug: How it works and how to avoid similar bugs. Hämtad från Synopsys: https://www.synopsys.com/blogs/software-security/heartbleed-bug/ (Hämtad 2023-05-22 kl. 11:45)
- **Aqua Security. (2023)**. Trivy. Hämtad från Trivy: https://aquasecurity.github.io/trivy/v0.34/ (Hämtad 2023-05-21 kl. 17:20)
- **Common Vulnerability Scoring System. (2023)**. Hämtad från FIRST.org: https://www.first.org/cvss/ (Hämtad 2023-05-20 kl. 08:55)
- **CRITICAL vulnerabilities in this report: CVE-2005-2541. (2025)**. Hämtad från National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2005-2541 (Hämtad 2023-05-27 kl. 10:05)
- **CRITICAL vulnerabilities in this report: CVE-2019-1010022. (2023)**. Hämtad från National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2019-1010022 (Hämtad 2023-05-26 kl. 15:40)
- **CRITICAL vulnerabilities in this report: CVE-2022-0543. (2023)**. Hämtad från National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2022-0543 (Hämtad 2023-05-26 kl. 09:10)
- **CRITICAL vulnerabilities in this report: CVE-2022-3734. (2023)**. Hämtad från National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2022-3734 (Hämtad 2023-05-25 kl. 13:55)
- **CRITICAL vulnerabilities in this report: CVE-2019-8457. (2023)**. Hämtad från National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2019-8457 (Hämtad 2023-05-22 kl. 16:25)
- **Download Node. (2023-05-26)**. Hämtad från Node.js: https://nodejs.org/en (Hämtad 2023-05-26 kl. 12:50)
- **ENISA. (2023)**. Statement on Log4Shell. Hämtad från ENISA: https://www.enisa.europa.eu/news/enisa-news/statement-on-log4shell/ (Hämtad 2023-05-23 kl. 07:30)

- **Gallo, K. (2022)**. Log4J Vulnerability Explained: What it Is and how to fix it. Hämtad från BuiltIn: https://builtin.com/cybersecurity/log4j-vulerability-explained (Hämtad 2023-05-20 kl. 14:15)
- **GeeksforGeeks. (2023)**. Microsoft Azure: Check for Apache Log4j vulnerability in Azure VMs. Hämtad från GeeksforGeeks: https://www.geeksforgeeks.org/microsoft-azure-check-for-apache-log4j-vulnerability-in-azure-vms/ (Hämtad 2023-05-25 kl. 09:35)
- **Kocher, L. (2023)**. 84 Percent Of Code Bases Have At Least One Open Source Vulnerability. Hämtad från Open Source For You: https://www.opensourceforu.com/2023/02/84-percent-of-code-bases-have-at-least-one-open-source-vulnerability/ (Hämtad 2023-05-23 kl. 18:40)
- **Mcbride, L. (2022)**. 5 Key Open Source Security Risks and How to Prevent them. Hämtad från Sonatype Blog: https://blog.sonatype.com/5-key-open-source-security-risks-and-how-to-prevent-them (Hämtad 2023-05-21 kl. 10:20)
- **Mchmarny. (2023)**. Vimp Repository. Hämtad från GitHub: https://github.com/mchmarny/vimp (Hämtad 2023-05-27 kl. 08:05)
- **Microsoft. (2023)**. What is Vulnerability Management? Hämtad från Microsoft Security: https://www.microsoft.com/en-us/security/business/security-101/what-is-vulnerability-management (Hämtad 2023-05-22 kl. 13:15)
- National Cyber Security Centre. (2023). Vulnerability Management. Hämtad från NCSC: https://www.ncsc.gov.uk/guidance/vulnerability-management (Hämtad 2023-05-21 kl. 19:45)
- **National Vulnerability Database. (2023)**. Hämtad från NIST: https://nvd.nist.gov/vuln-metrics/cvss (Hämtad 2023-05-25 kl. 16:55)
- **Octoverse. (2022)**. The state of open source software. Hämtad från GitHub Octoverse: https://octoverse.github.com/ (Hämtad 2023-05-24 kl. 10:30)
- **Plump, T. (2022).** Github's Octoverse report finds 97% of apps use open source software. Hämtad från VentureBeat: https://venturebeat.com/programming-development/github-releases-open-source-report-octoverse-2022-says-97-of-apps-use-oss/ (Hämtad 2023-05-23 kl. 22:00)
- **Snyk. (2023)**. What is Snyk? Hämtad från Snyk: https://snyk.io/product/ (Hämtad 2023-05-25 kl. 11:00)
- **TEAM ASCEND. (2019)**. The Five stages of Vulnerability Management. Hämtad från TeamAscend Blog: https://blog.teamascend.com/stages-of-vulnerability-management (Hämtad 2023-05-24 kl. 08:45)
- **WSL. (2023)**. Hämtad från Microsoft: https://learn.microsoft.com/en-us/windows/wsl/install (Hämtad 2023-05-24 kl. 16:10)

- **ZDNet. (2023)**. CISA orders federal agencies to mitigate Log4j vulnerabilities in emergency directive. Hämtad från ZDNet: https://www.zdnet.com/article/cisa-orders-federal-agencies-to-mitigate-log4j-vulnerabilities-in-emergency-directive/ (Hämtad 2023-05-22 kl. 19:25)
- **Özkan, E. (2022)**. What is Grype? Grype Usage with examples. Hämtad från Sysaix.com: https://sysaix.com/what-is-the-grype-grype-usage-with-examples (Hämtad 2023-05-26 kl. 07:50)

Bilaga 1

Snyk sårbarhetskanning. JSON objektets utformning:


```
{
      "id": "SNYK-DEBIAN11-APT-522585",
      "cpes": [],
      "title": "Improper Verification of Cryptographic Signature",
      "CVSSv3": "CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N",
      "credit": [
        ""
      ],
      "semver": {
        "vulnerable": [
          "*"
        ]
      },
      "exploit": "Not Defined",
      "patches": [],
      "insights": {
        "triageAdvice": null
      },
      "language": "linux",
      "severity": "low",
      "cvssScore": 3.7,
      "malicious": false,
      "isDisputed": false,
      "references": [
        {
          "url":
"https://security-tracker.debian.org/tracker/CVE-2011-3374",
          "title": "ADVISORY"
        },
        {
          "url":
"https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=642480",
          "title": "Debian Bug Report"
        },
        {
          "url":
"https://people.canonical.com/~ubuntu-security/cve/2011/CVE-2011-3374.h
tml",
          "title": "MISC"
```

```
        },
        {
          "url": "https://seclists.org/fulldisclosure/2011/Sep/221",
          "title": "MISC"
        },
        {
          "url": "https://snyk.io/vuln/SNYK-LINUX-APT-116518",
          "title": "MISC"
        },
        {
          "url": "https://ubuntu.com/security/CVE-2011-3374",
          "title": "MISC"
        },
        {
          "url":
"https://access.redhat.com/security/cve/cve-2011-3374",
          "title": "RedHat CVE Database"
        }
      ],
      "cvssDetails": [
        {
          "assigner": "NVD",
          "severity": "low",
          "cvssV3Vector":
"CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N",
          "cvssV3BaseScore": 3.7,
          "modificationTime": "2022-01-03T17:20:43.650545Z"
        }
      ],
      "description": "## NVD Description\n**_Note:_** _Versions
mentioned in the description apply only to the upstream `apt` package
and not the `apt` package as distributed by `Debian:11`._\n_See `How to
fix?` for `Debian:11` relevant fixed versions and status._\n\nIt was
found that apt-key in apt, all versions, do not correctly validate gpg
keys with the master keyring, leading to a potential man-in-the-middle
attack.\n## Remediation\nThere is no fixed version for `Debian:11`
`apt`.\n## References\n-
[ADVISORY](https://security-tracker.debian.org/tracker/CVE-2011-3374)\n
- [Debian Bug
Report](https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=642480)\n-
[MISC](https://people.canonical.com/~ubuntu-security/cve/2011/CVE-2011-
3374.html)\n-
[MISC](https://seclists.org/fulldisclosure/2011/Sep/221)\n-
```

```
[MISC](https://snyk.io/vuln/SNYK-LINUX-APT-116518)\n-
[MISC](https://ubuntu.com/security/CVE-2011-3374)\n- [RedHat CVE
Database](https://access.redhat.com/security/cve/cve-2011-3374)\n",
      "epssDetails": {
        "percentile": "0.51706",
        "probability": "0.00164",
        "modelVersion": "v2023.03.01"
      },
      "identifiers": {
        "CVE": [
          "CVE-2011-3374"
        ],
        "CWE": [
          "CWE-347"
        ],
        "ALTERNATIVE": []
      },
      "nvdSeverity": "low",
      "packageName": "apt",
      "creationTime": "2020-08-19T09:26:56.589605Z",
      "disclosureTime": "2019-11-26T00:15:00Z",
      "packageManager": "debian:11",
      "publicationTime": "2018-06-27T16:20:45.037549Z",
      "modificationTime": "2022-11-01T00:08:27.375895Z",
      "socialTrendAlert": false,
      "relativeImportance": "unimportant",
      "severityWithCritical": "low",
      "from": [
        "docker-image|docker.io/redis@latest",
        "apt/libapt-pkg6.0@2.2.4"
      ],
      "upgradePath": [],
      "isUpgradable": false,
      "isPatchable": false,
      "name": "apt/libapt-pkg6.0",
      "version": "2.2.4",
      "dockerBaseImage": "redis:7.0.11-bullseye"
    }
```

Bilaga 2

Trivy sårbarhetskanning. JSON objektets utformning:


```
{
          "VulnerabilityID": "CVE-2011-3374",
          "PkgID": "apt@2.2.4",
          "PkgName": "apt",
          "InstalledVersion": "2.2.4",
          "Layer": {
            "DiffID":
"sha256:8553b91047dad45bedc292812586f1621e0a464a09a7a7c2ce6ac5f8ba2535d
7"
          },
          "SeveritySource": "debian",
          "PrimaryURL": "https://avd.aquasec.com/nvd/cve-2011-3374",
          "DataSource": {
            "ID": "debian",
            "Name": "Debian Security Tracker",
            "URL":
"https://salsa.debian.org/security-tracker-team/security-tracker"
          },
          "Title": "It was found that apt-key in apt, all versions, do
not correctly valid ...",
          "Description": "It was found that apt-key in apt, all
versions, do not correctly validate gpg keys with the master keyring,
leading to a potential man-in-the-middle attack.",
          "Severity": "LOW",
          "CweIDs": [
            "CWE-347"
          ],
          "CVSS": {
            "nvd": {
              "V2Vector": "AV:N/AC:M/Au:N/C:N/I:P/A:N",
              "V3Vector":
"CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N",
              "V2Score": 4.3,
              "V3Score": 3.7
            }
          },
          "References": [
            "https://access.redhat.com/security/cve/cve-2011-3374",
            "https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=642480",
```

```
"https://people.canonical.com/~ubuntu-security/cve/2011/CVE-2011-3374.h
tml",
            "https://seclists.org/fulldisclosure/2011/Sep/221",

"https://security-tracker.debian.org/tracker/CVE-2011-3374",
            "https://snyk.io/vuln/SNYK-LINUX-APT-116518",
            "https://ubuntu.com/security/CVE-2011-3374"
          ],
          "PublishedDate": "2019-11-26T00:15:00Z",
          "LastModifiedDate": "2021-02-09T16:08:00Z"
        }
```

Bilaga 3

Grype sårbarhetskanning. JSON objektets utformning:


```
 {
   "vulnerability": {
    "id": "CVE-2005-2541",
    "dataSource":
"https://security-tracker.debian.org/tracker/CVE-2005-2541",
    "namespace": "debian:distro:debian:11",
    "severity": "Negligible",
    "urls": [
     "https://security-tracker.debian.org/tracker/CVE-2005-2541"
    ],
    "description": "Tar 1.15.1 does not properly warn the user when
extracting setuid or setgid files, which may allow local users or
remote attackers to gain privileges.",
    "cvss": [],
    "fix": {
     "versions": [],
     "state": "not-fixed"
    },
    "advisories": []
   },
   "relatedVulnerabilities": [
    {
     "id": "CVE-2005-2541",
     "dataSource": "https://nvd.nist.gov/vuln/detail/CVE-2005-2541",
     "namespace": "nvd:cpe",
     "severity": "High",
     "urls": [
      "http://marc.info/?l=bugtraq&m=112327628230258&w=2",

"https://lists.apache.org/thread.html/rc713534b10f9daeee2e0990239fa407e
2118e4aa9e88a7041177497c@%3Cissues.guacamole.apache.org%3E"
     ],
     "description": "Tar 1.15.1 does not properly warn the user when
extracting setuid or setgid files, which may allow local users or
remote attackers to gain privileges.",
     "cvss": [
      {
       "version": "2.0",
       "vector": "AV:N/AC:L/Au:N/C:C/I:C/A:C",
       "metrics": {
```

```
        "baseScore": 10,
        "exploitabilityScore": 10,
        "impactScore": 10
       },
       "vendorMetadata": {}
      }
     ]
    }
   ],
   "matchDetails": [
    {
     "type": "exact-direct-match",
     "matcher": "dpkg-matcher",
     "searchedBy": {
      "distro": {
       "type": "debian",
       "version": "11"
      },
      "namespace": "debian:distro:debian:11",
      "package": {
       "name": "tar",
       "version": "1.34+dfsg-1"
      }
     },
     "found": {
      "versionConstraint": "none (deb)",
      "vulnerabilityID": "CVE-2005-2541"
     }
    }
   ],
   "artifact": {
    "name": "tar",
    "version": "1.34+dfsg-1",
    "type": "deb",
    "locations": [
     {
      "path": "/usr/share/doc/tar/copyright",
      "layerID":
"sha256:8553b91047dad45bedc292812586f1621e0a464a09a7a7c2ce6ac5f8ba2535d
7"
     },
     {
      "path": "/var/lib/dpkg/info/tar.md5sums",
```

```
      "layerID":
"sha256:8553b91047dad45bedc292812586f1621e0a464a09a7a7c2ce6ac5f8ba2535d
7"
     },
     {
      "path": "/var/lib/dpkg/status",
      "layerID":
"sha256:8553b91047dad45bedc292812586f1621e0a464a09a7a7c2ce6ac5f8ba2535d
7"
     },
     {
      "path": "/var/lib/dpkg/status",
      "layerID":
"sha256:bee68ae43a83b10c9f490448abb719304af02a834f3557fda199c6e408ae8cc
7"
     },
     {
      "path": "/var/lib/dpkg/status",
      "layerID":
"sha256:df132c87bdb2ed2662fbcab6e9ecce353f6e8fe257797f442bc50bf70a78a08
9"
     }
    ],
    "language": "",
    "licenses": [
     "GPL-2",
     "GPL-3"
    ],
    "cpes": [
     "cpe:2.3:a:tar:tar:1.34+dfsg-1:*:*:*:*:*:*:*"
    ],
    "purl":
"pkg:deb/debian/tar@1.34+dfsg-1?arch=amd64&distro=debian-11",
    "upstreams": []
   }
  }
```

Bilaga 4

Samtliga unika sårbarheter enligt CVE-id efter vimp-skanning.

```
sqlite> SELECT DISTINCT exposure
FROM vul;
CVE-2005-2541
CVE-2007-5686
CVE-2007-6755
CVE-2010-0928
CVE-2010-4756
CVE-2011-3374
CVE-2011-3389
CVE-2011-4116
CVE-2013-4235
CVE-2013-4392
CVE-2016-2781
CVE-2017-11164
CVE-2017-16231
CVE-2017-18018
CVE-2017-7245
CVE-2017-7246
CVE-2018-20796
CVE-2018-5709
CVE-2018-6829
CVE-2019-1010022
CVE-2019-1010023
CVE-2019-1010024
CVE-2019-1010025
CVE-2019-19882
CVE-2019-20838
CVE-2019-8457
CVE-2019-9192
CVE-2020-13529
CVE-2020-16156
CVE-2021-33560
CVE-2021-36084
CVE-2021-36085
CVE-2021-36086
CVE-2021-36087
CVE-2022-0543
CVE-2022-0563
CVE-2022-1304
CVE-2022-29162
CVE-2022-3219
CVE-2022-3647
```

```
CVE-2022-3715
CVE-2022-3734
CVE-2022-48303
CVE-2022-4899
CVE-2023-0464
CVE-2023-0465
CVE-2023-0466
CVE-2023-25809
CVE-2023-27561
CVE-2023-28642
CVE-2023-29383
CVE-2023-29491
CVE-2023-31484
CVE-2023-31486
CVE-2023-0634
CVE-2022-24769
```