Mini Penetration Test – OWASP Juice Shop

Kurs: Ethical Hacking (DI6005) - Halmstad Högskola

Projekt: Planned Penetration Test

Handledare: Jonny Svensson

Grupp: Matilda Berndtsson, Lejla Dzanic, Sofija Kostic, Michelle Lander

Datum: 11-01-2026

1 Inledning

I takt med att allt fler tjänster samt verksamheter flyttas till webbaserade plattformar har informationssäkerhet blivit en ännu viktigare faktor för både organisationer och användare. Webbapplikationer hanterar ofta känslig information såsom användaruppgifter, betalningsinformation och andra systemfunktioner, vilket gör att de blir särskilt utsatta för attacker. Ett strukturerat och framför allt etiskt penetrationstest är därför ett bra verktyg för att identifiera säkerhetsbrister innan de hinner utnyttjas av otillåtna aktörer.

Den här rapporten syftar till att tillämpa teoretiska kunskaper i ett praktiskt projekt. Arbetet fokuserar på att planera och genomföra ett kontrollerat penetrationstest mot en medvetet sårbar webbapplikation.

2 Target Overview

Vi kommer att genomföra ett penetrationstest mot OWASP Juice Shop, vilket är en avsiktligt sårbar webbapplikation (single page application) som är skapat för utbildning och träning. Syftet med pentestet är att analysera och identifiera säkerhetsbrister i webbapplikationen i en kontrollerad testmiljö. För att säkerställa att testet gjordes i en isolerad testmiljö kördes OWASP Juice Shop lokalt med hjälp av Docker, som är en applikation soDetta penetrationstest har genomförts enligt etablerade ramverk som OWASP WSTG, vilket är ett ramverk som beskriver hur säkerhetstestning av webbapplikationer bör utföras på ett strukturerat och systematiskt sätt. Ramverket används för att identifiera vanliga säkerhetsbrister och säkerställa att testningen täcker relevanta områden (Loshin, 20022).

Följande komponenter omfattas av penetrationstestet:

Webbapplikationen OWASP Juice Shop

3 Omfattning (Scope)

3.1 In-scope

Penetrationstestet har begränsats till webbapplikationen som nås via den angivna URL:en. Det har inte genomförts skannig eller analys av bakomliggande infrastruktur såsom servrar, nätverk eller molnplattformar. Det som igår i testet är endast applikationsnivå (Layer 7 i OSI-modellen).

Det har varit fokus på applikationens funktionalitet och säkerhetsmekanismer. Detta inkluderar testning av autentisering och auktorisering, exempelvis hur användare identifieras, hur behörigheter tilldelas samt hur åtkomst till skyddade resurser hanteras.

Det ingick även testning av injektionsattacker, såsom SQL-injektion och Cross-Site Scripting (XSS), samt analys av accesskontroll för att identifiera eventuella brister i hur applikationen begränsar åtkomt till känsliga funktioner. Även säkerheten i applikationens API:er har analyserats, exempelvis hur data skickas mellan klient och server och hur dessa anrop kan manipuleras.

Testningen omfattar inte OSINT, digital forensik eller kryptografiska attacker. Det fokuserar i stället på kontrollerad och icke-destruktiv exploatering av säkerhetsbrister på applikationsnivå.

3.2 Out-of-scope

Tester som ligger utanför applikationsnivån har inte ingått i penetrationstestet. Detta omfattar bland annat nätverksgransking av bakomliggande infrastruktur, forensisk analys, social engineering samt överbelastningsattacker såsom Denial of Service (DoS).

Detta ingick inte i testet på grund av etiska, tekniska och tidsmässiga begränsningar samt eftersom de faller utanför projektets definierade syfte. Fokus för testningen har istället legat på säkerhetsbrister på applikationsnivå.

4 Rules of Engagement (RoE)

För att säkerställa att penetrationstestet genomförs på ett kontrollerat, etiskt och juridiskt korrekt sätt har tydliga regler för genomförandet definierats. Dessa regler beskriver vilka ramat som testningen sker inom, samt hur eventuella incidenter ska hanteras.

4.1 Legal authorization

Penetrationstestet genomförs enbart i den tillhandahållna OWASP Juice Shop-labbmiljön, Applikationen är avsiktligt sårbar och avsedd för utbildnings- och testsyfte, vilket innebär att testningen sker med fullständigt tillstånd. Testmiljön är simulerad och körs lokalt, vilket betyder att inga verkliga användare, produktionssystem eller att någon känslig data påverkas under testets genomförande.

4.2 Test Restrictions and Limitations

Testningen är begränsad till kontrollerad och icke-destruktiv säkerhetstestning. Fokus ligger på manuell identifiering och verifiering av sårbarheter på applikationsnivå, såsom autentisering, sessionshantering, accesskontroll och vanliga webbsårbarheter (exempelvis XSS och injektionsattacker).

Automatiserade attacker som kan påverka systemets tillgänglighet eller stabilitet genomförs inte, och testningen utförs enbart i den angivna testmiljön.

4.3 Prohibited activites

Följande aktiviteter är förbjudna inom ramen för penetrationstestet. Detta inkluderar överbelastningsattacker såsom Denial of Service (DoS), attacker som kan krascha eller försämra applikationens funktionalitet samt testning av system eller resurser utanför OWASP Juice Shop-miljön.

Manipulation eller radering av data som inte är en del av testsituationen genomförs inte, och inga försök görs att påverka applikationen på ett sätt som kan leda till permanent skada.

4.4 Communication

Eftersom testningen genomförs i en kontrollerad utbildningsmiljö, sker all kommunikation kring projektet internt inom gruppen samt med kursens handledare. Vid behov av vägledning eller vid osäkerhet kring testets omfattning kontaktas handledaren innan testningen fortsätter.

4.5 Incident handling

Om en sårbarhet eller oavsiktlig incident skulle upptäckas under testningen pausas testet omedelbart. Samtliga pågående aktiviteter avbryts för att förhindra ytterligare påverkan.

Incidenten dokumenteras genom att tidpunkt, tekniska detaljer, genomförda steg samt relevanta skärmdumpar sparas. Därefter informeras ansvarig handledare innan testningen återupptas. Testet fortsätter först efter godkännande, och incidenten sammanfattas i rapporten tillsammans med en bedömning av eventuell påverkan samt åtgärder.

5 Risk Bedömning

För att strukturera penetrationstestet och säkerställa att testningen fokuserar på de mest kritiska delarna av applikationen genomfördes en riskbedömning innan själva testningen påbörjades. Riskbedömningen syftar till att identifiera vilka tillgångar som är mest värdefulla inom applikationen samt att prioritera testaktiviteter baserat på potentiell påverkan.

5.1 Asset inventory

Inom ramen för testets omfattning identifierades flera centrala tillgångar som bedömdes vara särskilt viktiga. Användarkonton samt inloggningsuppgifter är en känslig tillgång, då ett intrång kan leda till kontokapning och obehörig åtkomst till applikationens funktioner.

Autentiseringstokens, i form av JSON Web tokens (JWT), identifierades som en annan viktig tillgång. Dessa används för sessionshantering och åtkomstkontroll. Manipulation av tokens kan möjliggöra eskalering av privilegier eller identitetskapning.

Administrativ funktionalitet utgör en högriskresurs eftersom den ger förhöjda rättigheter och möjlighet att kontrollera applikationen.

Vidare identifierades applikationens API-endpoints som viktiga tillgångar. Dessa hanterar backend-funktioner och databehandling. Om de skyddas på ett felaktigt sätt kan de exponera känslig information eller möjliggöra otillåten åtkomst. Slutligen inkluderar tillgångarna även känslig applikationsdata som kan visas genom felkonfigurationer, såsom i offentligt tillgängliga filer eller kataloger.

5.2 Prioriteringsplan

Testningen prioriterades utifrån en kombination av eventuell konsekvens vid intrång och sannolikheten för att en sårbarhet kan exploateras. Autentiserings- och auktoriseringsmekanismer testades först, eftersom brister inom dessa områden kan leda till obehörig åtkomst till både användar- och administratörskonton.

Därefter fokuserades testningen på accesskontroll till skyddade resurser, då svagheter här kan möjliggöra privilegieeskalering och åtkomst till känsliga funktioner. Inputvalidering och injektionssårbarheter analyserades i nästa steg, eftersom dessa kan leda till manipulation av data, eller påverkan på backend-system.

Sessionshantering och tokenhantering testades därefter för att identifiera risker kopplade till sessionskapning, tokenmissbruk eller identitetsimitation. Slutligen testades informationsläckage, som i många fall har lägre direkt påverkan, men kan bidra till att underlätta andra, mer avancerade, attacker.

6 LLM Appendix

6.1 Val av AI-verktyg

I detta projekt användes ChatGPT, ett Large Language Model (LLM)-baserat AI-verktyg utvecklat av OpenAI. Verktyget användes som ett stöd i inlärnings- och arbetsprocessen, främst för att underlätta förståelsen av penetrationstestningsmetodiken, tekniska begrepp och praktiska moment.

6.2 Prompts använda

AI användes som sagt som ett stöd i projektets genomförande. Fokus låg på att ställa frågor som hjälpte gruppen att förstå metodik, verktygsanvändning och tekniska detaljer kopplade till penetrationstestning.

Exempel på typer av prompts som användes inkluderar:

- Förklaringar av OWASP Web Security Testing Guide (WSTG)
- Förtydliganden kring hur olika specifika attacker fungerar, exempelvis SQL-injektion, XSS och hantering av JWT
- Hjälp med att förstå verktyg och teststeg

AI användes inte för att automatiskt genomföra attacker eller generera färdiga exploateringar.

6.3 Rå output från LLM

Rå output från AI-verktyg har inte använts direkt i den slutgiltiga rapporten. All information som erhölls från AI skrevs om med gruppens egna ord och verifierades genom praktisk testning i labbmiljön. AI fungerade alltså som ett stödjande hjälpmedel snarare än en primär källa.

6.4 Reflektion kring användning av AI

Användningen av AI var särskilt värdefull i detta projekt eftersom gruppen saknade tidigare praktisk erfarenhet av penetrationstestning. AI bidrog genom att förklara kommandon, hjälpa till med teststeg och annan verktygsanvändning, vilket gjorde det möjligt för gruppen att själva genomföra testerna.

7 Identifierade sårbarheter

1. **Login Admin**

   Beskrivning

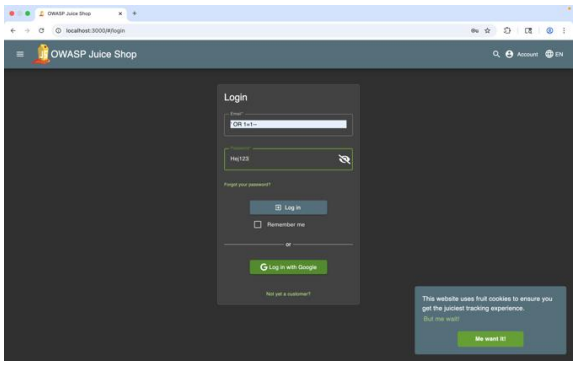
Applikationen är sårbar för SQL-injektionen i inloggningsfunktionen, genom manipulation i inloggningsfältet med SQL syntax så kunde autentiseringen kringgås utan giltiga inloggningsuppgifter.

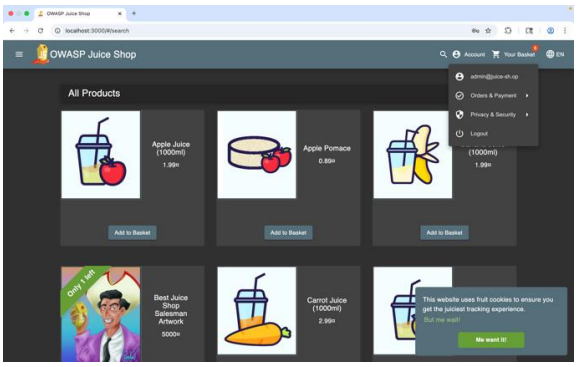
Påverkan

En obehörig angripare kan få åtkomst som inte ska vara tillkomlig förutom för användarkonton, detta inklusive administratörskonton, detta kan leda till dataläckage och full kontroll över applikationen.

Kommando

- Användarnamn: ‘ OR 1=1--
- Lösenord: valfritt





Mitigation

För att skydda säkerheten kan man implementera input-validering och sanering och även begränsa databasens rättigheter enligt principen om minsta privilegium.

2. **View Basket**

Beskrivning

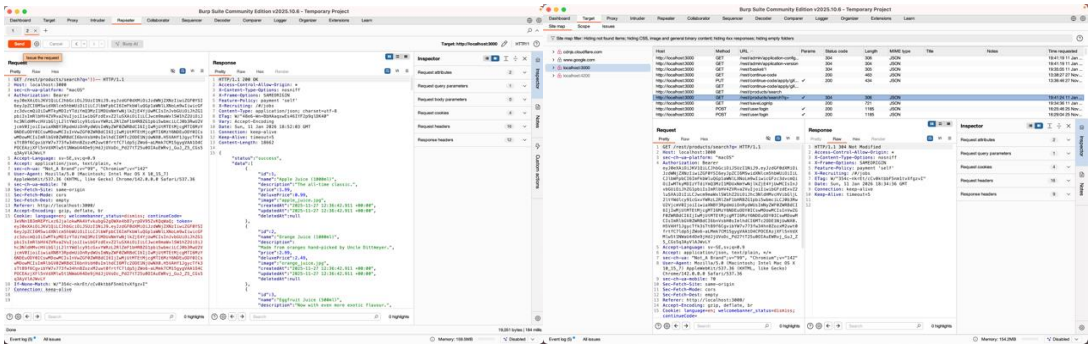
Applikationen har en funktion som spårar kundordrar via REST-API, denna saknar korrekt åtkomstkontroll som gör det möjligt för en autentiserad användare att komma åt andra användares orderinformation genom att manipulera order-ID i HTTP förfrågan.

Påverkan

Denna sårbarhet kan göra så att en angripare kan få tillgång till andra användares orderinformation, exponera känsliga uppgifter som orderstatus och beställningsdetaljer. Detta kan även bryta mot kund konfidentialitet.

Kommando

get /rest/track-order. ‘ GET /rest/track-order/5267-6f6928d0e342ac97' HTTP/1.1



Mitigation

För att åtgärda denna sårbarhet bör åtkomstkontroller på serversidan implementeras för att säkerställa att den begärda ordern tillhör den autentiserade användaren. Implementera konsekvent behörighetskontroll för alla API-endpoint och utföra regelbunden säkerhetstestning med fokus på Broken Access Control.

3. Product Search

Beskrivning

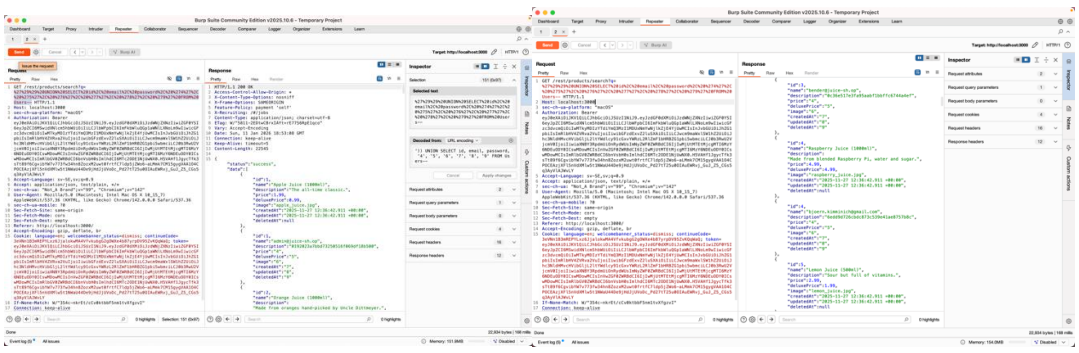
Applikationens sökfunktion för produkter är sårbar för SQL injektioner. Användarinmatning från sökfältet skickas direkt till backend-databasen utan tillräcklig validering. Detta gör det möjligt för angripare att manipulera SQL-frågan och få åtkomst till data som inte skall vara tillgänglig.

Påverkan

Denna sårbarhet har väldigt hög risk då den ger möjlighet att få full åtkomst till hela produktdatabasen, exponering av användardata och även inloggningsuppgifter. Även möjlighet till vidare attacker som kontokapning.

Kommando

GET /rest/products/search?q=%27%29%29%20UNION%20SELECT%20id%2C%20email%2C%20password%2C%20%274%27%2C%20%275%27%2C%20%276%27%2C%20%277%27%2C%20%278%27%2C%20%279%27%20FROM%20Users--
HTTP/1.1



Mitigation

För att förhindra denna typ av sårbarhet så kan man åtgärda det genom att använda parametriserade SQL-frågor, implementera strikt input-validering, begränsa databasens åtkomsträttigheter enligt princip om minsta privilegium och utföra regelbundna säkerhetstester med fokus på injektionsattacker.

4. **Login Jim and Bender**

Beskrivning

Användarinmatning i inloggningsformuläret hanteras bristfälligt och inkluderas direkt i SQL frågan utan korrekt validering eller användning av förberedda frågor. Detta gör att det blir möjligt att kringgå autentiseringsmekanismen. Genom att injicera SQL-syntax i användarnamnsfältet kan angriparen manipulera den underliggande SQL-frågan och logga in utan giltiga inloggningsuppgifter.

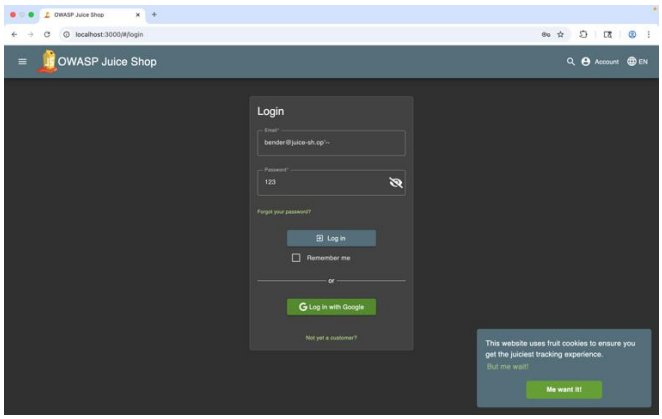
Påverkan

Denna sårbarhet möjliggör för obehörig inloggning på valfria användarkonton, fullständig kontokapning, åtkomst till användarspecifik information och funktionalitet och utföra åtgärder i offrets namn.

Kommando

Username: bender@juice-sh.op'--

Password: 123



Username: jim@juice-sh.op'--
Password: 123

Mitigation

För att åtgärda denna sårbarhet kan man implementera strikt validering och sanering av användarinmatning, undvika dynamiskt konstruerade SQL-frågor, införa skydd

mot brute force och inloggningsförsök och genomföra regelbundna säkerhetsgranskningar av autentiseringsfunktioner.

5. **Cross-Site Scripting (XSS)**

   Beskrivning
   Applikationen är sårbar för Cross-Site scripting, detta innebär att användarinmatning inte valideras korrekt innan den renderas i webbläsaren. Detta gör att en angripare kan injicera och exekvera godtycklig JavaScript kod i webbläsaren. Vid inmatning exekverades JavaScript koden i webbläsaren, vilket bekräftar XSS sårbarheten.

   Påverkan
   XSS sårbarheter kan utnyttjas för att stjäla sessioncookies, utföra åtgärder i användarens namn, omdirigera användare till skadliga åtgärder och även utföra phising attacker.

   Kommando
   <script>alert(1)</script>
   <iframe src=”javascript:alert(‘xss’)”>

   Mitigation
   För att förhindra dessa attacker så kan man implementera följande åtgärder: validera all användarinmatning, implementera content security policy, undvika inline javascript och använda etablerade ramverk med inbyggt XSS-skydd.

8 Etiska Överväganden

Penetrationstestet har genomförts med ett tydligt etiskt förhållningssätt och i en kontrollerad utbildningsmiljö. Ett medvetet val gjordes att utföra testet mot OWASP Juice Shop, som är en avsiktlig sårbar webbapplikation framtagen för utbildningssyfte. Därmed gjordes inga tester mot verkliga system och påverakde ingen arbetsmiljö, eller riktia användare eller verklig data. (OWASP, 2021).

Exploatering av identifierade sårbarheter skedde i syfte att verifiera dess existens och potentiella påverkan, inte för att orsaka skada. Trots att det är en testmiljö valdes det bort att göra överbelastningsattacker och dataraderingar. Vidare följdes principen att göra minsta möjliga påverkan, där endast utvalda teststeg utfördes för att uppnå förståelse för sårbarheterna. Resultaten dokumenterades på ett ansvarsfullt sätt och användes endast inom ramen för detta projekt. Detta förhållningssätt är i linje med etiska riktlinjer för penetrationstesting och informationssäkerhet. (OWASP, 2021; NIST, 2025).

En stor etisk frågeställning inom penetrationstestning och cybersäkerhet är om information och hackning, sårbarheter och angreppstekniker är öppet tillgängligt för allmänheten, trots risken att det kan användas i fel syfte. Ett begrepp som ofta används inom cybersäkerheten är

“you can’t defend what you can’t see” som även är en av kärnprinciperna hos OWASPs. Det bygger på idén att transparens och kunskap är avgörande för ett effektivt försvar. För att kunna skydda system måste försvarare förstå hur angripare tänker och agerar. (OWASP, 2021).

Forskning och ramverk inom informationssäkerhet visar att om kunskap kring hackning och sårbarheter hålls hemlig, kan kriminella förmodligen ändå få tillgång till det genom alternativa kanaler som forum och dark web. Detta gör att försvarssidan riskerar att sakna den kunskap som krävs för att förebygga och hantera säkerhetsincidenter. En riktlinje från NIST är tidigt identifiering och hantering av sårbarheter avgörande för att minska risken för intrång och annan skada. (NIST, 2025).

Pentester har en viktig roll i att stärka cybersäkerheten genom att identifiera sårbarheten innan de missbrukas. Detta bidrar till en ökad medvetenhet om sina säkerhetsbrister, att patchar och säkerhetsuppdateringar kan tas fram och etablera en långsiktig säkerhetskultur. Schneier betonar att säkerhet genom hemlighållande är ineffektivt, då angripare ofta redan känner till svagheterna, medan försvarare behöver samma kunskap för att skydda systemen. (Scheier, 2015).

Dagens utbildningar inom IT och informationssäkerhet lägger stor vikt på den etiska dimensionen av hackning. Idag ligger inte fokus på endast de tekniska färdigheterna men även på ansvar, lagstiftning och konsekvenserna av att missbruka sin kunskap. Detta projekt har följt dessa principer genom att använda tekniska verktyg och AI för lärande och förståelse, i stället för att automatisera attacker eller orsaka skada.

Sammanfattningsvis har penetrationstestet genomförts på ett etiskt försvarbart sätt, med fokus på ansvar, kunskap och skadeförebyggande arbete, i linje med etablerade principer för etisk penetrationstestning och öppen säkerhetsforskning. (OWASP, 2021; NIST 2012; Schneier, 2015).

9 Diskussion

Genomförandet av penetrationstestet mot OWASP Juice Shop gav en inblick i hur teoretiska koncept kan tillämpas i praktiken. Eftersom ingen i gruppen hade tidigare erfarenhet av praktisk penetrationstestning var det svårt att inledningsvis sätta i gång, särskilt vad gäller verktygsanvändning. Det var viktigt att förstå hur webbapplikationens arkitektur fungerade, samt hur klient och server kommunicerade via API:er, för att kunna hitta relevanta attackytor.

En av de största utmaningarna under projektet var att skilja mellan vad som är en faktisk sårbarhet och var som endast är en funktion eller avsiktlig design i applikationen. Eftersom OWASP Juice Shop är byggd för utbildningssyfte innehåller applikationen medvetet osäkra funktioner. För att säkerställa att identifierade problem faktiskt var sårbarheter krävdes förståelse för applikationens funktionalitet. Sårbarheten kollades i sitt sammanhang för att avgöra om den var en verklig säkerhetsrisk eller endast var en del av applikationens funktionella uppbyggnad.

Identifierade sårbarheter såsom SQL-injektion, XSS och bristande accesskontroll visade tydligt att "enkla" misstag kan ge omfattande konsekvenser. Detta understryker vikten av grundläggande säkerhetsprinciper, såsom korrekt inputvalidering och server-side accesskontroller. Testerna gav också insikt i hur kombinationer av flera mindre brister kan leda till mer avancerade attacker, exempelvis kontokapning eller exfiltrering av känslig data.

Etiken i arbetet var ständigt närvarande. Även om testmiljön var kontrollerad var det viktigt att exploatera sårbarheter med minimal påverkan och dokumentera resultaten noggrant. Detta speglar den balans som professionella penetrationstestare måste upprätthålla mellan praktisk nyfikenhet och ansvar. Diskussionen om varför säkerhetsinformation är öppet tillgängligt, trots potentiellt missbruk, visade att öppenhet och kunskap är avgörande för ett effektivt försvar. Man kan alltså inte skydda det man inte förstår.

Slutligen gav projektet värdefulla insikter kring förbättringar för framtida testning. Bland annat skulle en mer noggrann och systematisk dokumentation av alla teststeg och tydligare koppling mellan sårbarheter och OWASP top 10 ha varit fördelaktigt. Även tidsplanering för olika testfaser kan förbättras för att optimera arbetet.

10 Referenser

Loshin, P. (2022, 3 Mars). *Open Web Application Security Project (OWASP)*. TechTarget. https://www.techtarget.com/searchsoftwarequality/definition/OWASP

NIST (2025). Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile (SP 800-61 Rev. 3).

https://csrc.nist.gov/pubs/sp/800/61/r3/final

OWASP (2021) PWASP Top 10 – web application security risks.

https://owasp.org/Top10/2025/

Schneier, B. (2015) Data abd Goliath. W.W. Northon & Company.

Appendix

OWASP WSTG

Steg 1 – Informationsinsamling

- Vilka funktioner?

- Login
- Register
- Search
- Produktsidor
- Korgen
- Feedback/kontakta

- Inspect → Network i webbläsaren. Ladda om sidan. Vi ska nu kunna se:
  - Vilka endpoints finns?
  - Vilka API:er?
  - Hur data skickas
  - Hur login fungerar

- Använd Burp Suite:
  1. Starta Burp Suite.
  2. Gå in på start-menyn
  3. Sök efter Burp Suite Community Edition
  4. Starta programmet
  5. Klicka “Temporary Projct”, “Next”, “Use Burp defaults”, “Start Burp” (Burp kommer öppnades med flera flikar)
  6. Klicka på fliken “Proxy”, klicka “Proxy Settings”, och under “Proxy Listeners” ska det stå “Interface: 127.0.0.1:8080”. Det betyder att proxyn är aktiverad.
  7. Ställ in webbläsaren så att den använder Burps proxy: Installera FoxyProxy Standard, (FoxyProxy - Chrome Web Store), klicka “FoxyProxy-ikonen” och gå till “Option”. Klicka på “Add New Proxy”. Fyll i “Title” = Burp, “Proxy Type” = HTTP, “Proxy IP” = 127.0.0.1, “Port” = 8080. Klicka Save och aktivera proxyn i webbläsaren genom att klicka på dens ikon i och sedan välja “Use proxy “Burp” for all URLs, eller “BURP (Enabled)”.
  8. Starta interception
  9. Klicka runt på sidan (för nu fångas all trafik upp)

- Samla in information om
  - Cookies
  - Tokens (JWT)
  - Headers
  - Serverinformation
  - API-rutter
  - Parametrar
  - Forms och inputfält
  - Javascript-filer (kolla för dolda funktioner)

Steg 2 – Identifiera attackytor

- Var finns inloggningen?
- Var finns formulär?
- Var skickas data till API?
- Var kan man skriva text?
- Var används ID:n i URL:er?
- Leta efter:
  - Inmatningsfält

- Endpoints med parametrar
- API-kall som går att manipulera
- Sessions cookies
- Headers som kan ändras
- Admin-paneler
- Känsliga sidor

Burpsuite repeater search field

Leta upp en GET /rest/products/search?q= i burpsuite, man kan bara söka på typ “juice” i search field. Skicka till repeater och fyll i ‘))-- efter q. då får man upp alla produkter på hemsidan.

Denna får ut alla saker att köpa. (Och användare) (Man ska lägga det efter q= i repeater)
GET
/rest/products/search?q=%27%29%29%20UNION%20SELECT%20id%2C%20email%2C%20password%2C%20%274%27%2C%20%275%27%2C%20%276%27%2C%20%277%27%2C%20%278%27%2C%20%279%27%20FROM%20Users-- HTTP/1.1

Detta ovanför får ut konton som Jim & bender. Man kan sedan logga in på dessa konton genom o göra SQL injection med ‘ samt --.

Logga in som bender
Username: bender@juice-sh.op’--
Password: a

Logga in som jim
Username: jim@juice-sh.op’--
Password: 123

Installera docker för lättare pentesting: https://www.docker.com/products/docker-desktop/
Efter att ha gett tillåtelse se till att den fungerar, gå till command prompt/terminal och skriv: “docker --version”
Sen, kör Juice Shop (i terminalen):
docker pull bkimminich/juice-shop (den här raden laddar ner juice shop som en Docker-image från internet till datorn)
docker run –p 3000:3000 bkimminich/juice-shop (den här startar Juice shop)
Sen, gå till “http://localhost:3000“(körs lokalt då, och alla attacker borde fungera)

http://localhost:3000/#/score-board - På denna sidan så listar de deras attackytor och sårbarheter

Vi kör med Windows och i den vanliga Terminalen