Utveckling och säkerhetsutvärdering av en flexibel autentiseringslösning

EMIL JÖNSSON & ANDREAS NILSSON
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY |
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Bachelor's Thesis

Utveckling och säkerhetsutvärdering av en flexibel autentiseringslösning

Av

Emil Jönsson & Andreas Nilsson

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Sammanfattning

Detta examensarbete har utförts i samarbete med Knowit Experience i Malmö och har haft som huvudsyfte att utvärdera och utveckla en autentiseringslösning till en webbhandelsplattform.

Arbetet inleddes med ett antal semistrukturerade intervjuer med mjukvaruutvecklare på företaget som gav förståelse för deras behov. Med bakgrund av dessa gjordes en utvärdering av de två autentiseringstjänsterna Auth0 och Okta med avseende på kostnad och användarvänlighet. Okta valdes som tjänst. Intervjuerna var också grunden till en kravspecifikation. Med bakgrund av kravspecifikationen designades en modul med hjälp av klassdiagram och sekvensdiagram. Modulen utvecklades sedan agilt.

Examensarbetet producerade en ASP.NET-modul som möjliggör konfiguration för autentisering med hjälp av en godtycklig OpenID Connect-tjänst. En initial säkerhetstestspecifikation producerades även baserad på OWASP Web Security Testing Guide. Slutligen utvärderades webbapplikationen med hjälp av denna testspecifikation.

Nyckelord

OAuth2, OpenID Connect, EPiServer, elicitering, penetrationstest.

Abstract

This thesis work was done in collaboration with Knowit Experience in Malmö. Its main goal was to evaluate and develop an authentication module for the company's e-commerce platform.

Semi-structured interviews were held with software developers at the company to better understand their needs. Based around these needs an evaluation regarding price and usability was conducted for the authentication services Auth0 and Okta. Okta was chosen for implementation. The interviews also produced a requirements specification. This specification was used to design a module using class diagrams and sequence diagrams. The module was then developed using agile methods.

This thesis work produced an ASP.NET module that generalizes configuration for authentication using an arbitrary OpenID Connect service. A security test specification based on OWASP Web Security Testing Guide was also produced. Finally, the application was evaluated using this test specification.

Keywords

OAuth2, OpenID Connect, EPiServer, elicitation, penetration test.

Förord

Vi vill tacka Knowit Experience i Malmö för samarbetet, i synnerhet våra handledare Mathias Andersson och Per Nergård för all hjälp och ett varmt välkomnande under vårt distansarbete. Vi vill även passa på att tacka vår handledare Ben Smeets och vår examinator Christian Nyberg på Lunds Tekniska Högskola.

Emil Jönsson och Andreas Nilsson

Innehåll

Terminologi

Anspråk - ett påstående om en entitet med tillhörande metadata. Anspråk används ofta för att mappa användaregenskaper till rättigheter för en resurs (Exempel. *namn, e-postadress*). (eng. *claim)*

Autentisering - att verifiera identiteten hos en entitet (oftast en användare).

Användaragent - Mjukvara som kan hämta, rendera och facilitera interaktion mellan slutanvändare och webbinnehåll. (eng. *user-agent*)

Resursägare - en entitet som kan ge tillgång till en skyddad resurs. I kontexten av detta arbete, systemets slutanvändare (administratörer eller kunder i systemet). (eng. *resource owner*)

Resursserver - servern som hostar skyddade resurser. Servern kan svara på resursförfrågningar med användning av access tokens. I kontexten av detta arbete, EPiServer CMS. (eng. *resource server*)

Klient - en applikation som ber om tillgång till skyddade resurser från resursservern, å resursägarens vägnar. I kontexten av detta arbete, företagets webbapplikation. (eng. *client*)

Authorization code - en kod som ges till användaren som sedan används av resursservern för att hämta ut en access token från autentiseringsservern.

Access token - en token utfärdad av autentiseringsservern efter lyckad autentisering. Används av resursservern för att kontrollera om en användare är inloggad och dess rättigheter. Innehåller dess levnadstid och eller andra egenkonfigurerade fält.

Refresh token - en token utfärdad av autentiseringsservern efter lyckad autentisering. Används för att hämta en ny access token efter att denna har gått ut utan att användaren behöver omautentisera.

ID-token - en token utfärdad av autentiseringsservern efter lyckad autentisering. Denna innehåller anspråk om en användare som kan

användas för att hålla koll på om en användare blivit autentiserad eller övrig intressant om en användare som t.ex. namn eller.

Auktoriseringsserver - en server som tillhandahåller access tokens och refresh tokens till klienten efter lyckad autentisering av resursägare. I kontexten av detta arbete, den valda autentiseringstjänsten. (eng. *authorization server*)

Autentiseringsmodul - den modul examensarbetarna ska utveckla till företagets webbapplikation.

MFA - multifaktoriell autentisering.

Social inloggning - Inloggning där sociala medier utför autentiseringen av en identitet, till exempel inloggning via Google eller Facebook.

1 Inledning

1.1 Bakgrund

Knowit är ett konsultbolag som erbjuder digitala lösningar för tre affärsområden, Experience med inriktning mot frontend och användbarhet, Insight för managementkonsulter och Solutions för backend och systemutveckling. Knowit grundades år 1990 och har idag ca. 2700 medarbetare i Sverige, Danmark, Norge, Finland och Tyskland. Detta arbete utfördes på Knowit Experience i Malmö.

I ett internt projekt utvecklar Knowit en webbhandelsplattform, i denna rapport kallad webbapplikation, baserad på det kommersiella publiceringssystemet EPiServer. Applikationen var i behov av en autentiseringslösning som var lättförståelig för användare och enkel att konfigurera för utvecklare. Säkerhet och användarvänlighet stod i centrum. Företaget efterfrågade stöd för social inloggning och eventuellt MFA. Därutöver fanns det krav på autentiseringslösningen att hantera olika användartyper och tilldela dessa olika roller. I detta arbete var dock den primära typen av användare tänkt att vara en framtida webbhandlare.

Applikationen använde tekniker som ASP.NET, React och TypeScript och implementationen modulariserades så att den kunde återanvändas i andra liknande applikationer.

1.2 Syfte

Syftet med examensarbetet var att utvärdera tre autentiseringstjänster till Knowits existerande webbapplikation och komma fram till vilken av dessa som var bäst lämpad baserat på deras krav på säkerhet, flexibilitet och användarvänlighet.

1.3 Mål

Examensarbetet utvärderade olika autentiseringstjänster till Knowits existerande webbapplikation och valde ut den som var mest lämplig baserat på teknik, funktion och kostnad. Med kostnad var det viktigt att beakta

denna både ur Knowits perspektiv samt slutkundens. Detta resulterade i en modul som kunde autentisera användare med stöd för social inloggning. Målet för arbetet var att implementera en modul i form av en Minimal Viable Product (MVP) i en existerande webbapplikation som var baserad på EPiServer-plattformen för att slutligen säkerhetstesta modulen med en egenutvecklad testspecifikation.

1.4 Problemformulering

Under examensarbetet besvarades följande frågor:

1. Om det finns någon, hur ser den nuvarande autentiseringslösningen ut?
2. Vilka fördelar prioriteras under utvärderingen?
3. Hur ska autentiseringsalternativen utvärderas?
4. Vilket autentiseringsalternativ är mest lämplig för företagets nuvarande webbapplikation?
5. Hur ska en *användarvänlig* autentiseringslösning implementeras?
6. Hur ska en *flexibel* autentiseringslösning implementeras?
7. Hur ska en *säker* autentiseringslösning implementeras?

1.5 Motivering av examensarbetet

Vi valde detta examensarbete då webbutveckling är ett område vi båda kan tänka oss arbeta med i framtiden. Därutöver lockade det oss då examensarbetet gav oss möjligheten att utveckla en lösning och inte enbart var en undersökande studie.

Knowit var i behov av en modulär och flexibel autentiseringslösning, inte enbart för detta projekt, utan även för framtida projekt av liknande natur. En utredning av de alternativ som finns på marknaden hade hjälpt företaget bestämma vad som passar dem bäst kostnads- och funktionsmässigt.

Examensarbetet kändes även relevant för vår samtid. Då digitalisering av samhällstjänster, handel och underhållning blir allt vanligare, blir kraven på internetsäkerhet allt hårdare. En inventering av olika lösningsalternativ som finns på marknaden, samt deras kvaliteter och brister, främjar därför inte enbart uppdragsgivaren, utan även hela IT-branschen och därmed samhället i stort.

1.6 Avgränsningar

Vi valde att avgränsa analysen till endast tre autentiseringslösningar eftersom vi ville kunna utvärdera dessa tillräckligt. Det var också bra att begränsa analysen eftersom vi efter analysen även skulle fullständigt implementera en av dessa lösningar. Implementationen gjordes med färdiga autentiseringstjänster och skulle därför inte utvecklas från grunden av oss. Under utvecklingen räckte det att en implementation av en av lösningarna gjordes.

Då EPiServer är baserat på ASP.NET MVC-ramverket fokuserades framtagningen av autentiseringsmodulen till detta ramverk. EPiServer hade under arbetets gång ett pågående betaprogram för ASP.NET Core, men stöd för detta utreddes inte i detta arbete.

Implementationen implementerades med hjälp av de officiella anvisningarna för den valda autentiseringstjänsten i den mån det var möjligt. Eventuella säkerhetsbrister utreddes och kommenterades.

Säkerhetstest som gjordes under arbetet var baserade på OWASPs testlista, men ett flertal tester valdes bort. Arbetet ämnade testa säkerheten hos den valda autentiseringstjänsten, företagets webbapplikation som är byggd ovanpå EPiServer och integrationen mellan autentiseringstjänsten och webbapplikationen. Tester från OWASPS testlista som närmare angränsade till dessa tre områden var därför prioriterade i detta arbete.

2 Teknisk bakgrund

Under detta examensarbete användes ett flertal verktyg och tekniker. Dessa beskrivs kort nedan.

2.1 EPiServer CMS

EPiServer, numera Optimizely, är ett webbpubliceringssystem (*content management system*). Med hjälp av dess API kan en användare bygga och förändra hemsidors innehåll direkt i ett webbläsargränssnitt. I API:n ingår även databasintegration vilket illustreras i Bild 1 nedan. EPiServer CMS använder Microsofts ASP.NET-webbramverk [1].

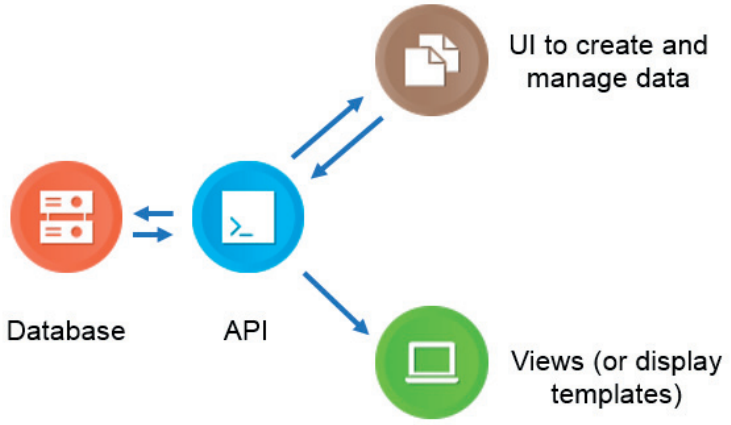
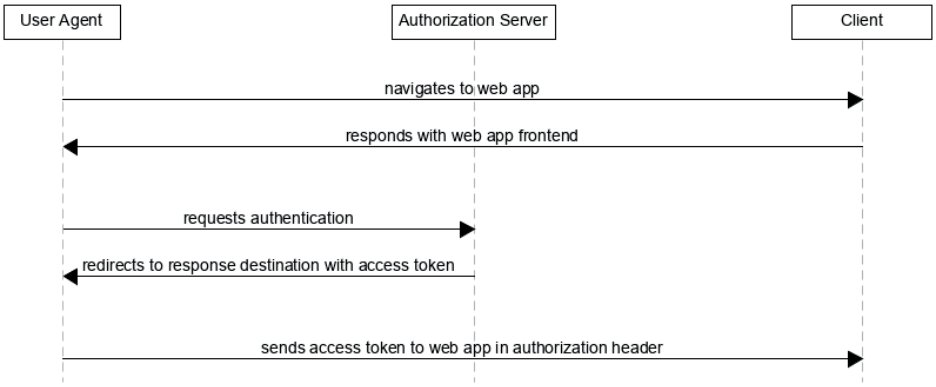


Bild 1. *Överblicksbild av EPiServers utformning.*
Källa: Tagen från [1]

2.2 OAuth 2.0

OAuth 2.0 är ett öppet autentiseringsprotokoll som är standardiserad av IETF och används för att tillåta en tredjepartsapplikation att få begränsad tillgång till en HTTP-tjänst, antingen genom att skapa en förbindelse mellan resursägaren och HTTP-tjänsten eller tillåta tredjepartsapplikationen att få tillgång till en resurs själv [2].

Istället för att använda resursägarens egna autentiseringsuppgifter för att komma åt skyddade resurser, sättet som autentisering gjorts på sedan internets födelse, får klienten en av standarden definierad access token. Denna token består av en sträng innehållandes bland annat accessrättigheter och dess livslängd. Denna utfärdas av en auktoriseringsserver och kan användas till att implicit autentisera och auktorisera klienten så att denna kan komma åt resurser hostade på en annan server. Standarden definierar även en token kallad refresh token. Denna token har oftast en längre livslängd och används för att hämta nya access tokens efter att deras livslängd har gått ut. Det finns ett antal tjänster som implementerar OAuth2-protokollet, däribland tjänsterna Auth0, Keycloak och Okta som kommer bli relevanta i detta arbete.



Figur 1. *Ett typiskt OAuth2-flöde.*

2.3 JSON Web Token (JWT)

JSON Web Tokens är en öppen standard av IETF för att representera säkerhetsanspråk mellan två parter. Dessa tokens innehåller anspråk som ett JSON-objekt och tillåter dessa att integritetsskyddas med en Message Authentication Code (MAC) eller någon form av digital signatur. [3]

2.4 OpenID Connect

OpenID Connect (OIDC) är ett tunt identitetslager som utökar OAuth2-protokollet. Med detta lager kan klienters identitet verifieras på ett enhetligt sätt av en autentiseringsserver. Lagret inför en JWT vid namn

ID-token. Denna token innehåller anspråk om en användare som kan användas för autentisering [4].

ID-tokens bevarar sin integritet genom att signeras enligt standarden JSON Web Signature (JWS). Därtill kan de erhålla konfidentialitet genom att krypteras i enlighet med standarden JSON Web Encryption (JWE) [5]. Kryptering är dock inte obligatorisk.

2.5 OWASP

The Open Web Application Security Project, eller förkortat OWASP är en ideell organisation vars mål är att förbättra säkerheten för världens mjukvara [6]. OWASP består av tiotusentals medlemmar som arbetar med opensource-projekt och har sedan 20 år tillbaka skapat verktyg, guider och föreläsningar för att förbättra utvecklares säkerhetstänk. De har bland annat tagit fram ett 400-sidorsdokument vid namn OWASP Web Security Testing Guide där de beskriver ett hundratal olika typer av attacker mot mjukvara och vad utvecklare kan göra för att testa och skydda mot detta.

2.6 OWASP Zed Attack Proxy

OWASP Zed Attack Proxy (ZAP) är ett program som kan användas till att analysera och hitta eventuella brister i en webbapplikation [7]. Programmet är utvecklat av OWASP-organisationen och är ett av de verktyg som kommer att användas under säkerhetsutvärderingen i detta arbete.

2.7 Burp Suite

Burp Suite är ett program för att upptäcka och analysera sårbarheter i säkerheten hos webbsidor och webbapplikationer [8]. Verktyget erbjuder en mängd olika funktioner som automatisk skanning av förfrågningar och responser, navigera i webbapplikationer via en proxy samt möjligheten att utföra replayattacker och analysera sessionstokens. Programmet är ett av de verktyg som kommer användas i säkerhetsutvärderingen i detta arbete.

2.8 Nmap

Nmap är en gratis och öppen programvara som kan användas för att analysera säkerhetsaspekter i nätverk och webbapplikationer [9]. Programmet är ett av de verktyg som kommer användas i säkerhetsutvärderingen i detta arbete.

2.9 Wireshark

Wireshark är ett program som kan användas för att logga och analysera nätverksprotokoll samt den ingående och utgående datan på ens nätverk [10].

3 Metod

Examensarbetet omfattar elicitering av Knowits autentiseringsbehov och analys av befintliga tjänster för att hitta den mest lämpliga för deras webbapplikation. Det omfattar även design, utveckling, funktionell testning och säkerhetstestning av en implementation av den valda tjänsten.

Arbetet delades upp i följande faser: Elicitering, inläsning och utvärdering av autentiseringstjänster, design, utveckling, funktionell utvärdering och säkerhetsutvärdering. Samtliga faser gjordes enligt agil metod och överlappade därför i tid. Följande uppdelning gjordes till förmån för rapportens struktur.

Projektet organiserades enligt den agila projektmodellen Kanban. Ett annat alternativ hade kunnat vara *Scrum* men denna valdes bort med bakgrund av att arbetet inte skulle behöva förhålla sig till sprints och iterationer. Examensarbetarna definierade listorna *Backlog*, *Design, To-Do*, *Doing*, *Code Review*, *Testing* och *Done*. För att illustrera och organisera arbetsflödet användes Trello som digitalt verktyg.

3.1 Elicitering

Eliciteringen utfördes delvis för att ställa krav på den autentiseringstjänst som skulle väljas och delvis för att utreda vilken funktionalitet autentiseringsmodulen skulle innehålla.

Lauesen förespråkar att intervjuer utförs med andra än officiella representanter och gärna från flera olika användargrupper [11]. Projektet var ett in-house projekt i tidig utveckling och inga faktiska användare fanns ännu. Då funktionalitet främst berörde hantering och förvaltning av autentiseringsmodul och autentiseringstjänst bedömdes handledarna på företaget vara lämpliga intervjuobjekt, då de var utvecklare för webbapplikationen eller annars vana e-handelsutvecklare.

Svaren dokumenterades i ett intervjudokument (se bilaga 1). De krav som formulerades efter elicitering dokumenterades i form av Trellokort. Dessa kort tilldelades unika ID-nummer och testinstruktioner för att verifiera att kraven uppfylldes under projektets gång. Kraven sammanställdes slutligen i

en kravspecifikation under kategorierna *Autentiseringstjänst* och *Autentiseringsmodul* (se bilaga 2).

3.1.1 Elicitering av autentiseringstjänst

Inledningsvis gjordes en ad-hoc marknadsundersökning för att identifiera olika autentiseringstjänster som skulle platsa i utvärderingen. Efter detta hölls ett eliciteringsmöte med handledare på företaget för att utreda deras behov och komma överens om vilka autentiseringstjänster som skulle jämföras under arbetet. I denna fas kom även företagets definition på *flexibel* och *användarvänlig* att utredas. Elicitering gjordes fortlöpande under arbetets gång i samband med veckoliga avstämningsmöten.

3.1.2 Elicitering av autentiseringsmodul

Elicitering av den modul som skulle utvecklas bestod av semistrukturerade intervjuer och öppna samtal med handledare på företaget. Fortsatt elicitering gjordes genom demonstration av utvecklingsframstegen i samband med veckoliga avstämningsmöten.

3.2 Inläsning och utvärdering av autentiseringstjänsterna

Inläsning gjordes genom att läsa de tre tjänsternas officiella dokumentation och instruktioner samt leta reda på övrig hjälp via t.ex internetforum. Därefter utvecklades en enklare prototyp för vardera autentiseringstjänst för att enklare förstå hur de fungerade.

Utvärderingen av autentiseringstjänsterna skedde genom att ett dokument upprättades där varje tjänst jämfördes utefter de kvaliteter som eliciteringssvaren gav (se bilaga 3). Med bakgrund av detta presenterades resultatet för Knowit där de sedan bestämde sig för vilken av de tre tjänsterna de vill ha implementerad i sin webbapplikation.

3.3 Design

Projektets designområden omfattade för det första design av den autentiseringsmodul som skulle utvecklas. För det andra omfattade det även

design av den specifikation som ämnade testa den färdiga implementationens säkerhet.

3.3.1 Design av autentiseringsmodul

En design på autentiseringsmodulen togs fram i form av ett klassdiagram. Modulens interaktion med den valda autentiseringstjänsten och företagets webbapplikation illustrerades med sekvensdiagram för handlingarna **inloggning** och **hämtning av skyddade resurser**.

3.3.2 Design av testspecifikation

Testerna designades och genomfördes för autentiseringstjänsten, företagets webbapplikation och den mellanliggande integrationen. Dessa tester utformades enligt *OWASP Web Security Testing Guide v 4.2* [12]. Ett urval av tester gjordes i enlighet med projektets avgränsning och dokumenterades i projektets testspecifikation (se bilaga 4).

Testerna bestod av manuella tester i webbläsare, dels av automatiserade och manuella tester med hjälp av *Nmap*, *OWASP Zed Attack Proxy (ZAP)* och *Burp Suite Community Edition*. Vid testning i webbläsare användes *Google Chrome* som primär webbläsare och *Mozilla Firefox* som sekundär webbläsare vid behov av mer än en samtidig session. Vissa tester innefattade granskning av nätverkstrafik i *Wireshark*.

3.4 Utveckling

Utvecklingen utfördes i enlighet med de steg som examensarbetarna definierat i sin Kanbanmodell. Koden versionshanterades genom användning av git och bitbucket. På bitbucket utvecklades arbetet på en egen branch på företagets repository och kunde fortlöpande delas med företaget. Kodgranskning av samtlig kod som skrevs under arbetet utfördes av en handledare på företaget för att försäkra att kodens kvalitet och effektivitet höll företagets standard.

Utveckling gjordes i *Visual Studio 2019 Community Edition* och modulen programmerades i C# med ramverket *ASP.NET MVC*.

Kontinuerlig kodgranskning utfördes i Kanbansteget *Code Review* av den examensarbetare som inte ansvarat för Trellokortet. Slutgiltig kodgranskning utfördes av handledare på Knowit.

3.5 Funktionell utvärdering

Kontinuerlig funktionell utvärdering utfärdes av examensarbetarna under projektets gång i Kanbansteget *Testing*. Fördelaktigen, men inte tvunget, utfördes testet av examensarbetaren som inte ansvarat för motsvarande Trellokort.

Den slutgiltiga funktionella utvärderingen skedde genom en presentation för Knowit. I samband med denna hölls ett acceptansmöte. Under detta möte kom alla funktionskrav att verifieras så att handledarna var försäkrade om att dessa var uppfyllda. I slutet av mötet godkändes eller underkändes lösningen.

3.6 Säkerhetsutvärdering

Utvärderingen av säkerhet gjordes av examensarbetarna. Utförandet av testerna skedde enligt testspecifikationen som togs fram under testdesignen (se bilaga 4).

Vid utförande av vissa test granskades systemet efter exponering av känslig data. Information om den autentiserade användaren A som inte var *e-postadress*, *förnamn, efternamn* eller *guid* ansågs falla under detta begrepp. Samtliga användarvariabler för användare A som exponerades för en annan autentiserad användare B ansågs också vara känslig data.

En översiktlig presentation av resultatet görs i denna rapport (se kapitel 4.6). Detaljerade resultat av säkerhetsutvärdering sammanställdes i ett test-resultatsdokument (se bilaga 5).

3.7 Kommunikation

Kommunikationen under arbetets mellan examensarbetarna och Knowit skedde på distans på grund av covid-19-pandemin. Samtal mellan examensarbetare skedde med hjälp av Voice over IP-programmet Discord

där båda hade möjligt att skärmdela sitt arbete för att kunna hjälpa och visa varandra kod och funktionalitet. För kommunikation med Knowit användes Slack för textkommunikation och Microsoft Teams för avstämningsmöte som hölls måndag.

3.8 Anpassningar under arbetets gång

Arbetets mål ändrades under dess gång. Från början var arbetet ämnat att jämföra tre olika autentiseringstjänster. Under inläsning och elicitering identifierades Auth0, Okta och Keycloak. Den sistnämnda uteslöts dock ur jämförelsen efter ett tidigt eliciteringsmöte med Knowit. Eftersom företaget letade efter en autentiseringstjänst där de själva eller kunden inte behövde ansvara för hostning var Keycloak inte ett relevant alternativ. Följaktligen lades ingen vidare tid på att jämföra tjänsten med Auth0 och Okta.

En annan förändring av metoden under arbetets gång var hur mycket fokus som lades på jämförelsen av de två kvarstående tjänsterna, Auth0 och Okta. Detta för att utvärderingen visade snabbt att tjänsterna var mycket lika i sin implementation (se kapitel 4.2). Under examensarbetets gång beslöt sig Okta dessutom för att köpa upp Auth0 [13]. Eventuella framtida ihopslagningar av tjänsterna hade gjort jämförelsen irrelevant och fokus ändrades till att göra en kortare jämförelse och en mer utbredd säkerhetsgranskning.

Slutligen anpassades autentiseringsmodulens krav mot slutet av arbetet. Istället för att låta Okta sköta rollhanteringen, valde Knowit att detta skulle hanteras med EPiServers API. Följaktligen ströks de modulkrav som relaterade till rollhantering i Okta.

3.9 Källkritik

I detta delavsnitt motiveras och kommenteras de valda källornas tillförlitlighet.

Vad gäller säkerhetsaspekter användes primärt inte EPiServer, Okta eller Auth0 som källor. Istället användes internetstandarder från IETF eller riktlinjer från OWASP som primär källa då dessa är ideella och därmed bedöms vara mindre partiska.

EPiServer
EPiServer är ett svenskt företag som ligger bakom webbpubliceringssystemet med samma namn. Företaget har en mängd betydande kunder i Norden, både inom den privata och offentliga sektorn.

Som skapare av systemet som arbetets webbapplikation bygger på, innehar de systemets officiella dokumentation och relevanta utvecklingsguider.

Detta gäller källa [1] och [24].

Internet Engineering Task Force (IETF)
IETF är en underorganisation till Internet Society (ISOC), och består av en löst sammansatt grupp av bland annat nätverksutvecklare och forskare. Organisationen formulerar frivilliga standarder för Internet. De har bland annat samarbetat med Google i utvecklingen och lanseringen av transportprotokollet QUIC.

Deras mål är att få Internet att fungera bättre genom att producera relevanta tekniska dokument som håller hög kvalitet.

Detta gäller källa [2], [3] och [5].

OpenID Foundation
Denna organisation förvaltar OpenID Connect-teknologin och dess community. De utfärdar även certifikat för organisationer som använder deras teknologi.

Flera stora aktörer stödjer OpenID Foundations verksamhet, däribland *Ebay, Google, Microsoft* och *PayPal.*

Detta gäller källa [4] och [19].

The OWASP Foundation
The Open Web Application Security Project är en ideell organisation som arbetar för att förbättra säkerheten i webb-baserad hårdvara. Stiftelsen anordnar öppna mjukvaruprojekt och har tiotusentals medlemmar över hela världen.

Flera stora aktörer stödjer OWASPs verksamhet, däribland *Atlassian, Adobe* och *Oracle.*

Detta gäller källa [6], [7] och [12].

PortSwigger

PortSwigger är skaparna av Burp Suite, som är ett av de program som OWASP rekommenderar som verktyg för säkerhetsutvärdering och anses därför vara pålitligt.

Detta gäller källa [8].

Nmap

Nmap är ett av de program som OWASP rekommenderar som verktyg för säkerhetsutvärdering och anses därför vara pålitligt.

Detta gäller källa [9].

Wireshark

Wireshark är ett av de program som OWASP rekommenderar som verktyg för säkerhetsutvärdering och anses därför vara pålitligt.

Detta gäller källa [10].

Software Requirements: Styles and Techniques

Boken är författad av Soren Lauesen — professor emeritus på IT-universitetet i Köpenhamn.

Boken används som kurslitteratur för kursen Kravhantering, ETSF30 på Lunds Tekniska Högskola.

Detta gäller källa [11].

TechCrunch

TechCrunch är en internettidning som fokuserar på tekniknyheter. De har cirka 17 miljoner månatliga läsare, har bedömts som *most reliable* enligt Ad Fontes Media [14] och sponsras av företag som *Adobe, Dell Technologies* och *Toyota.*

Detta gäller källa [13] och [20].

Ad Fontes Media
Ad Fontes Media är en organisation som granskar nyheter utefter pålitlighet och politisk bias. Deras uppdrag är att göra nyhetskonsumenter smartare och nyhetsproducenter mer sakliga.

Detta gäller källa [14].

Okta
Okta är skaparna av plattformen *Okta*, en av de ledande autentiserings-plattformerna på marknaden.

Detta gäller källa [15] och [17].

Auth0
Auth0 är skaparna av plattformen *Auth0*, en av de ledande autentiserings-plattformerna på marknaden.

Detta gäller källa [16] och [18].

OWIN
OWIN är ett öppet källkodsprojekt som är betrott av *Microsoft.*

Detta gäller källa [21].

Per Ignatius
P. Ignatius har en kandidatexamen i datateknik vid Högskolan Dalarna. Han examensarbete är godkänt av högskolan och har publicerats i den svenska databasen för vetenskapliga artiklar.

Detta gäller källa [22].

Cybersecurity Ventures

Detta företag är skapare av tidskriften *Cybersecurity Magazine*. Deras artiklar har stor spridning och citeras frekvent i stora tidskrifter som *Wall Street Journal*, *New York Times, USA Today* och *Huffington Post*.

Detta gäller källa [23].

4 Resultat

I detta kapitel presenteras de resultat som arbetets olika faser producerade.

4.1 Elicitering

Kraven sammanställdes i dokumentform och presenteras i bilaga 2, där genomstrukna krav representerar krav som under utvecklingens gång utgick.

4.1.1 Eliciteringsresultat av autentiseringstjänst

Företagets definition på en *flexibel* autentiseringslösning visade sig innebära att denna skulle vara konfigurerbar för utvecklare samt förvaltare av applikationen. Konfigurationshandlingar som ändring av en användares grupptillhörighet, anspråk, namn och användning av MFA skulle således vara möjliga direkt i administratörsgränssnittet. Den skulle även vara anpassningsbar till företagets liknande webbapplikationer och integrerbar i andra autentiseringstjänster än den som slutligen valdes av företaget.

Företagets definition på en *användarvänlig* autentiseringslösning var att användaren ska ha möjlighet att logga in med hjälp av autentiseringsmekanismer kopplade till populära sociala medier som Google och Facebook.

Mötet belyste också att kostnad var en av punkterna som skulle jämföras hos autentiseringstjänsterna. Tjänsterna borde även erbjuda en gratisplan med få begränsningar och få skillnader mellan gratis och betald version. På så vis förväntades utvärderingen av gratisplanen kunna ge en mer exakt förhandsvisning av den betalda versionen.

Tjänsterna Auth0 och Okta valdes som de två autentiseringsalternativ som under arbetet skulle komma att jämföras. Keycloak valdes bort med bakgrund av företagets önskan att inte behöva ansvara för hostning av tjänsten själva. Dessutom ansåg både företag och examensarbetare att det vore lämpligare att utveckla och jämföra två prototyper istället för tre, med tanke på arbetsbelastningen. Analys och jämförelse av Auth0 och Okta sammanställdes och presenteras i bilaga 3.

4.1.2 Eliciteringsresultat av autentiseringmodul

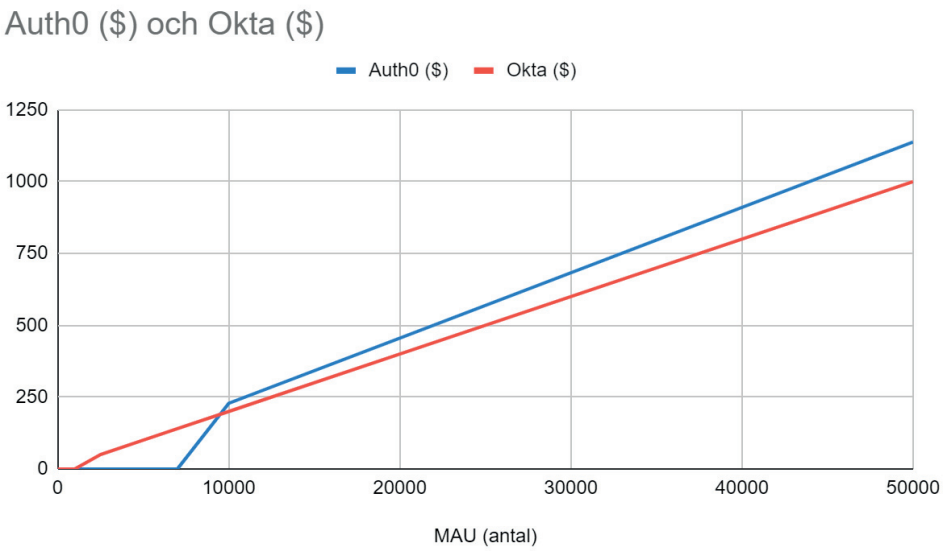
En undersökning av webbapplikationens befintliga autentiseringslösning visade att denna utgick från en standardlösning för autentisering i EPiServers mallprojekt Alloy, med tillagd användning av ASP.NETs Identity-ramverk. Kunder kunde inte själva registrera sig utan detta kunde endast ske via EPiServers administratörsgränssnitt.

Denna lösning bedömdes vara otillräcklig. Anpassningsmöjligheterna för funktioner som social inloggning och MFA var skrala. Autentiseringen skedde via ett webbformulär som självt inte hade stöd någon av dessa funktioner. Webbapplikationen var således i behov av en annan lösning för autentisering för att möjliggöra introduktion av social inloggning och för att självregistrering för kunder skulle kunna utföras på ett enkelt och säkert sätt. Ingen del av det ursprungliga användargränssnittet för autentisering skulle bevaras.

4.2 Inläsning och utvärdering av autentiseringstjänsterna

En prototyp av varje autentiseringstjänst utvecklades i webbapplikationen med hjälp av protokollet OpenID Connect. Efter inläsning av internetstandarder och granskning av liknande lösningar för autentisering på internet bestämdes det att OpenID Connect och OAuth2 skulle användas för att utveckla modulen.

Båda tjänsterna hade lösningar för flera typer av MFA och stöd för social inloggning, däribland populära tjänster som Microsoft, Facebook och Google. Detta påvisade att båda var möjliga alternativ för Knowits webbapplikation. Eftersom kostnaden var en viktig faktor för Knowit undersöktes de två tjänsternas priser i korrelation till månatliga aktiva användare (eng. *monthly active users*, MAU). Nedan i figur 2 ses respektive tjänsts prisplan i förhållande till användarantal.



Figur 2. *Kostnadsjämförelse mellan Auth0 och Okta, före införandet av Oktas nya gratisplan.*

Användbarhetstester som utfördes mellan Auth0 och Okta gjordes genom att räkna antal knapptryck det tog för att navigera från respektive autentiseringstjänsts administratörstartsida till önskat mål. Funktioner som testades var bland annat *konfigurering av identity provider, konfigurering av MFA, skapande av användargrupper* och *konfigurering av en grupps anspråk*. Dessa undersökningar säkerställde att båda hade ett enkelt navigerbart gränssnitt för att en person med teknisk bakgrund inte skulle stöta på problem när denna ska utföra de vanligaste tänkbara handlingarna för att hantera användare. En sammanställning av antalet klick som examensarbetarna behövde för att utföra dessa handlingar presenteras i tabell 1 nedan. Ingen betydande skillnad påträffades mellan tjänsterna. För vidare kommentarer angående användarvänlighet, se bilaga 3.

Handling	**Antal klick**	
	Auth0	**Okta**
Lägga till eller konfigurera identity provider	6	5

Lägga till MFA	4	5
Konfigurera till MFA	4	6
Skapa en användargrupp	4	4
Konfigurera en användargrupps anspråk	5	5
Aktivera CORS / Trusted Origins	5	6

Tabell 1. *Antal klick för utförande av typiska konfigurationshandlingar.*

En mindre granskning av båda tjänsternas säkerhet gjordes vilket visade att båda ansågs säkra nog för att användas i Knowits applikation. Dels på grund av att deras rykte och samarbete med välkända företag (se [15] och [16]) och dels på grund av att deras implementation av OAuth2 och OpenID Connect var gjord enligt respektive standard (se [17] och [18]). Båda tjänsterna har certifierats av OpenID Foundation [19]. Den enda skillnaden arbetet kom i kontakt med gällande säkerhet i tjänsterna var, efter granskning av bådas administratörsgränssnitt, att Auth0 saknade revokering av access tokens. Detta bedömdes dock inte som en säkerhetsbrist eftersom detta kunde kompenseras genom att ge access tokens kortare livslängd.

En vidare säkerhetsundersökning för autentiseringstjänsterna uteblev på grund av tidsbrist och Oktas köp av Auth0 [14]. Istället lades fokus på att säkerhetsutvärdera en fullständig implementation av den, av företaget, valda tjänsten.

Utvärderingen dokumenterades i ett gemensamt dokument och skickades till företaget (se bilaga 3).

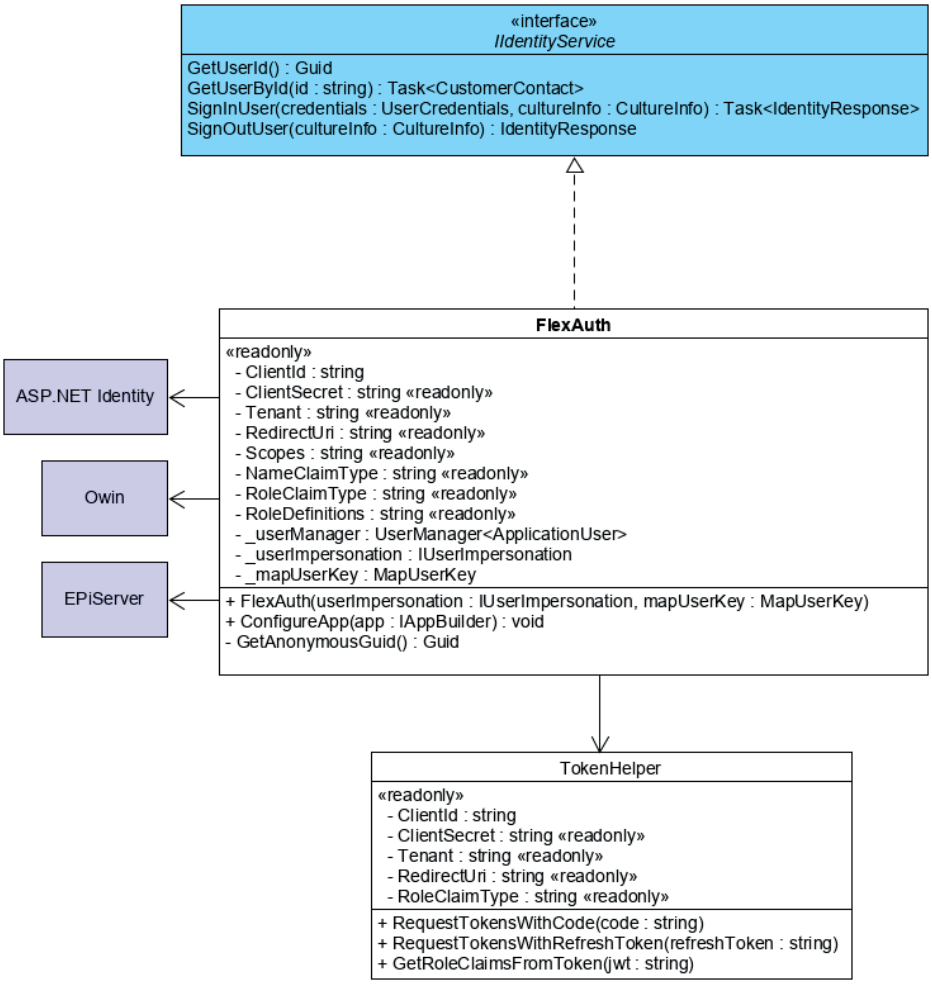
Knowit valde Okta som autentiseringstjänst med bakgrund av det som sammanställdes i jämförelsedokumentet. Oktas köp av Auth0 och införandet av en gratisplan för Okta med bättre villkor [20] påverkade också valet. Med arbetet mer fokuserat på en tjänst påbörjades arbetet med autentiseringsmodulens design.

4.3 Design

En första tanke angående autentiseringslösningens design var att denna skulle vara enligt *Resource Owner Password Flow*. Detta skulle innebära en vidareutveckling av den gamla autentiseringslösningen där en användare loggar in med användarnamn och lösenord direkt i webbappen. Backenden skulle därefter skicka denna information till autentiseringstjänsten varpå denna svarar med en access token. Dock valdes inte detta flöde eftersom det vara en säkerhetsrisk enligt OAuth2-standarden och bör bara användas då inga andra flöden är möjliga enligt sektion 4.3 i [2]. Designen planerades därför implementera *authorization code flow* vilket skulle betyda att en användare skulle bli omdirigerad till autentiseringsservern för att autentisera sig för att sedan skickas tillbaka till webbapplikationen. Efter samtal med företaget bedömdes omdirigering för autentisering vara acceptabelt för användarupplevelsen om inloggningsskärmens utseende kunde anpassas. *Authorization code flow* kom sedan att ändras till *Hybrid flow* på grund av utvecklingsbegränsningar (se kapitel 4.4).

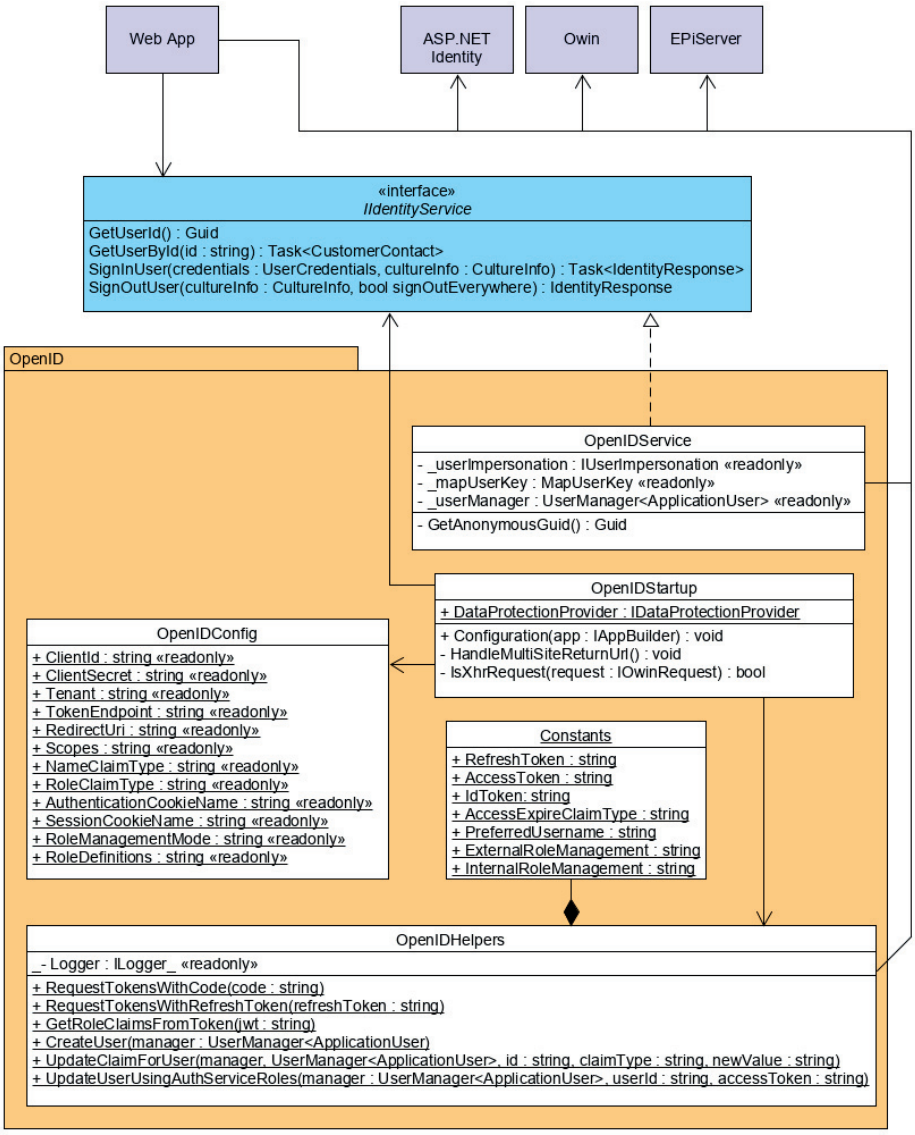
4.3.1 Designresultat av autentiseringsmodul

För att visualisera autentiseringsmodulens design upprättades ett klassdiagram som redovisar modulens klasser, attribut, funktioner och dess relationer till webbapplikationen och relevanta ASP.NET-moduler. Den första versionen bestod av två klasser. Den första klassen *FlexAuth* ansvarade för alla autentiseringsanrop och uppsättning med OWIN, ett öppet webbgränssnitt för webbapplikationer [21]. Den andra klassen *TokenHelper* interagerade med autentiseringstjänstens tokenslutpunkt och hade hjälpmetoder för att hantera JWTs. Klassdiagrammet visas i figur 3 nedan. Under utveckling visade det sig att *FlexAuths* många ansvarsområden orsakade ett beroendeproblem i webbapplikationen. Detta reviderades i den andra designiterationen.



Figur 3. *Klassdiagram över första designversionen.*

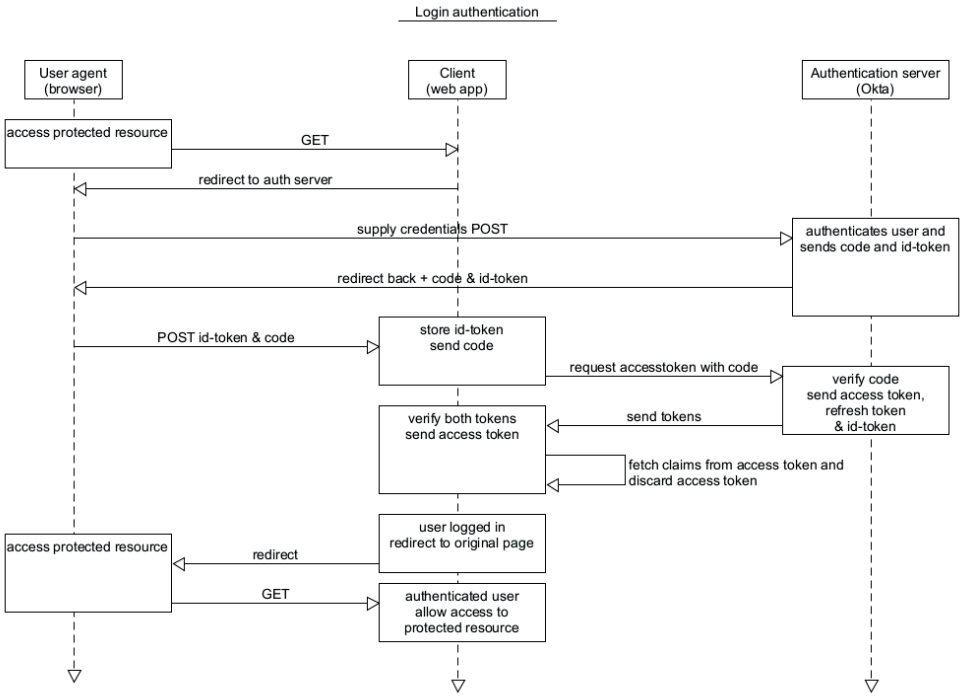
Under utvecklingens gång gick designen genom en ytterligare iteration i takt med att förståelsen för protokollen och det befintliga systemet ökade. Den monolitiska klassen *FlexAuth* bröts upp i klasserna *OpenIDStartup, OpenIDService, OpenIDConfig* och *OpenIDHelpers* med en intern *Constants*-klass. Detta löste beroendeproblemen från första design-iterationen som uppstod på grund av *FlexAuth*-objektets livstid. Som resultat av detta blev koden även tydligare uppdelad. Designen från denna iteration presenteras i figur 4 nedan.



Figur 4. *Klassdiagram över den slutgiltiga versionen av designen.*

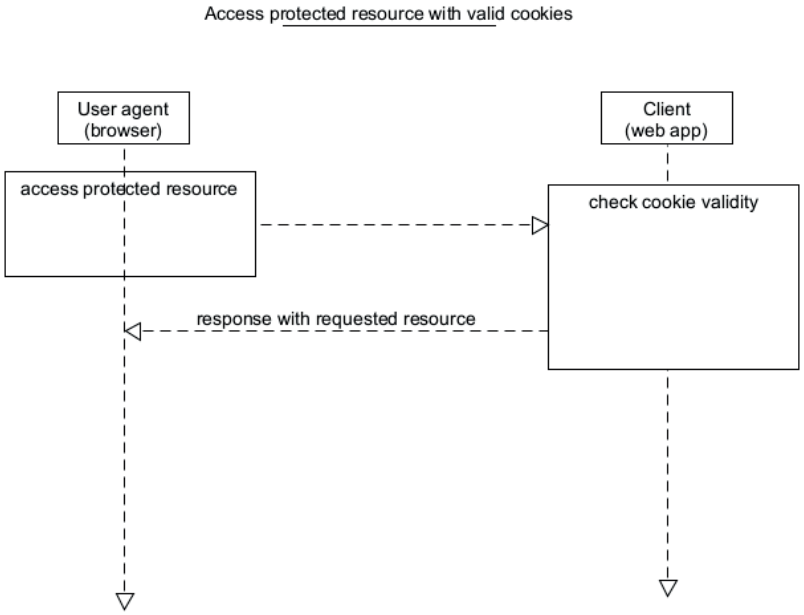
Följande sekvensdiagram designades för att kunna planera flödet mellan de tre viktigaste parterna under autentiseringen: webbläsare (User agent), webbapplikation (Client) och autentiseringsserver (Okta).

Flödet för inloggning och autentisering inleds med att en användare ber webbapplikationen om en skyddad resurs varpå denne omdirigeras till autentiseringsservern. Användaren är antingen redan inloggad i Okta eller behöver ange användarnamn och lösenord. Efter lyckad autentisering ger autentiseringsservern användaren en kod och en ID-token. Dessa skickas sedan till webbapplikationen för att användas för hämtning av refresh token och access token. Efter detta skapas en autentiseringscookie för att spara den inloggade användaren i webbläsaren. Denna cookie kan sedan användas för att hämta en skyddad resurs, t.ex. användarens varukorg. Autentiseringsflödet illustreras i figur 5 nedan.

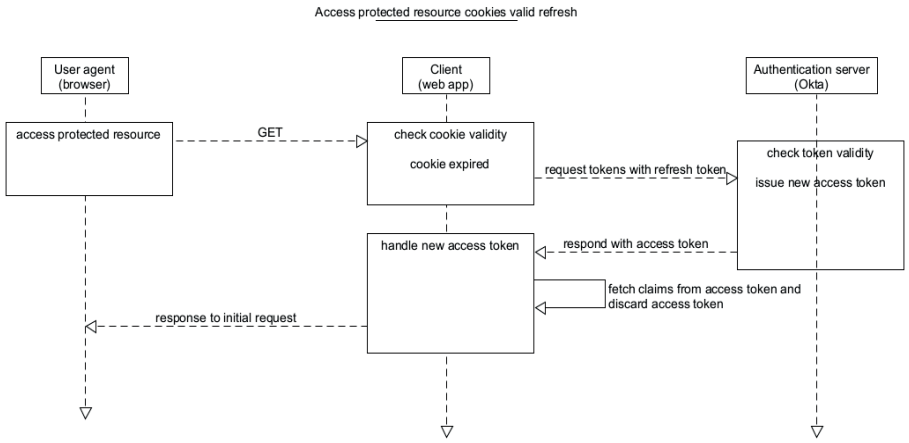


Figur 5. *Autentiseringsflöde mellan användare, webbapplikationen och autentiseringsservern vid inloggning.*

När en användare redan är inloggad och har en giltig autentiseringscookie kan denna komma åt alla skyddade resurser denna är behörig att få tillgång till. Flödet för en lyckad respektive misslyckad hämtning av en skyddad resurs illustreras i figur 6 respektive 7.

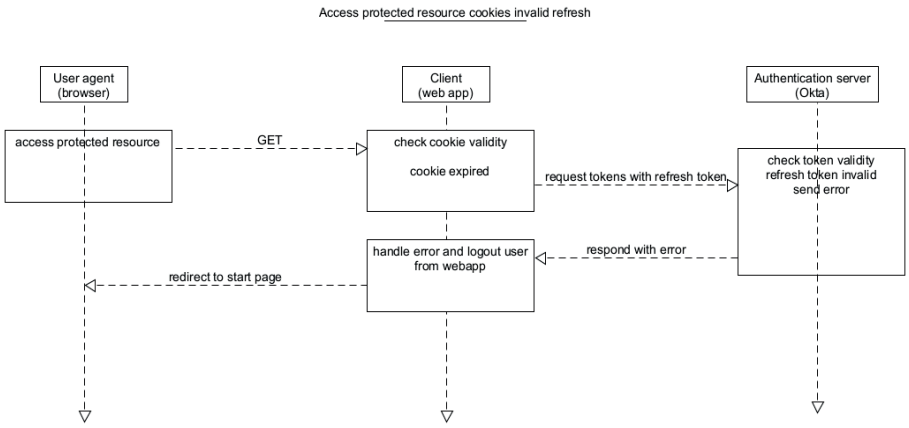


Figur 6. *Autentiseringsflöde mellan användare, webbapplikationen och autentiseringsservern vid åtkomst av skyddad resurs där användaren har giltiga cookies.*



Figur 7. *Autentiseringsflöde mellan användare, webbapplikationen och autentiseringsservern vid åtkomst av skyddad resurs där användaren inte har giltiga cookies.*

Om användaren inte har en giltig refresh token när autentiseringscookien ska uppdateras kommer användaren att loggas ut. Detta illustreras i figur 8 nedan.



Figur 8. *Autentiseringsflöde mellan användare, webbapplikationen och autentiseringsservern vid åtkomst av skyddad resurs där användaren inte har giltiga cookies och refresh token har utgått.*

4.3.2 Designresultat av testspecifikation

Under modulens utveckling blev det tydligare vilka säkerhetsaspekter som skulle testas och utvärderas. Då autentiseringsmodulen i sin grund ämnar lösa verifiering och hantering av identiteter valdes kategorierna *Identity Management Testing* och *Authentication Testing*. Modulen skulle även samspela med EPiServer, ASP.NET Identity och OWIN för att hantera rättigheter och sessioner, varpå *Authorization Testing* och *Session Management Testing* inkluderades i utvärderingen. Slutligen valdes *Testing For Weak Cryptography* och *Client-side Testing* för att inkludera en initial utvärdering av företagets webbapplikation och dess konfiguration. På så vis kunde eventuella säkerhetsbrister i applikationen identifieras. Därutöver kunde riktlinjer för konfiguration och installation definieras, för att sedan agera hjälpdokument inför applikationens lansering.

Följaktligen bortprioriterades testkategorierna *Information Gathering, Configuration and Deploy Management Testing, Data Validation Testing, Error Handling, Business Logic Testing* och *API Testing*. Utöver dessa

kategorier uteslöts även *WSTG-ATHZ-01* ur kategorin *Authorization Testing* från specifikationen, då detta tidigare testats i ett tidigare arbete av Per Ignatius, där han utvärderade säkerheten i en applikation baserad på identisk teknologi [22]. Således designades testscenarion för 44 av 97 OWASP-tester.

4.4 Utveckling

Utvecklingen skedde enligt metoden i kapitel 3.4 och en fullt fungerande modul producerades och konfigurerades som sedan delades med företaget via den webbaserade lagringstjänsten *bitbucket*. Okta-kontot som användes under prototyputvecklingen konfigurerades för användning vid testning och demonstration. I bild 2 nedan presenteras en lätt anpassad inloggnings-skärm som användes i modulen.

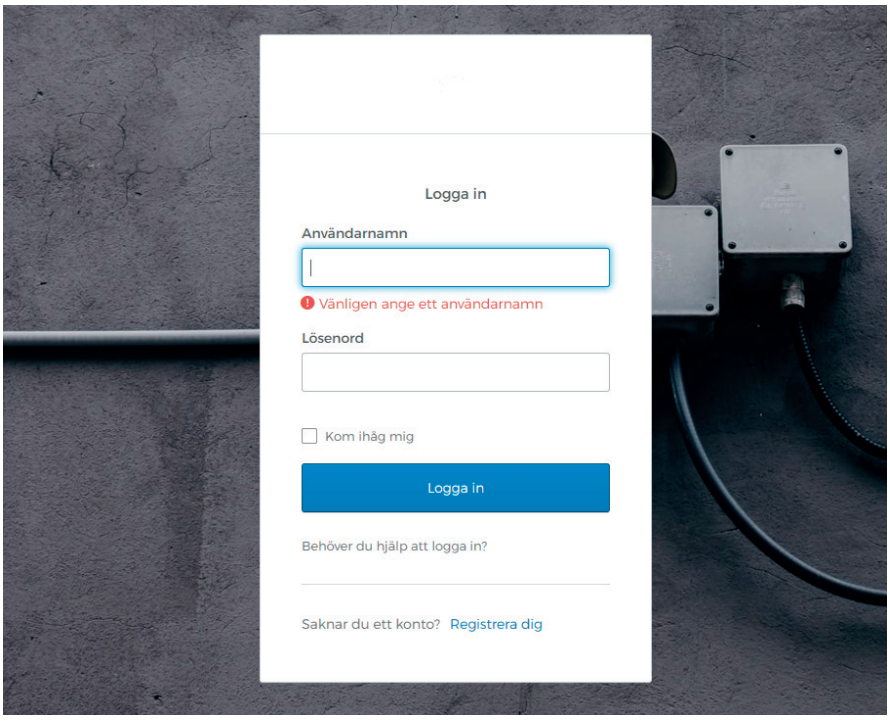


Bild 2. *Resultatbild från Oktas inloggningsskärm.*

Med bakgrund av inläsning av både OAuth2 och OpenID Connect samt säkerhetstips från både Okta och Auth0s guider utvecklades modulen enligt kravspecifikationen. Utvecklingen under arbetet var till stor del test- driven

och drevs av tester från både kravspecifikationen och säkerhetstest-specifikationen.

Autentiseringen planerades i designfasen att göras med OAuth2-flödet *authorization code flow* vilket är det rekommenderade sättet att hantera access tokens på med avseende på säkerhet enligt sektion 1.3.1 i [2]. Paketet som användes för OpenID Connect-autentisering i OWIN, webbgränssnittet för applikationen, visade sig dock inte ha en utlösande händelse vid användning av *authorization code flow*. Lösningen blev att istället implementera *Hybrid flow* som har specificerats av OpenID Connect under sektionen "Authentication using the Hybrid Flow" i [4]. Detta innebar att en ID-token även gavs direkt i samband med lyckad autentisering. Då en användares eventuellt stulna ID-token inte ensamt kan presenteras för att autentisera sig och få tillgång till applikationen, ansågs detta vara en acceptabel lösning. Således kunde applikationen undvika att använda *implicit flow*, som ökar exponeringen av access tokens och därmed anses vara ett av de mer osäkra flödena enligt standarden enligt sektion 10.3 [2].

4.5 Funktionell utvärdering

Samtliga funktionskrav verifierades fortlöpande av examensarbetarna genom kravens tillhörande tester. Autentiseringsmodulen demonstrerades för företaget för att dokumentera kravens slutgiltiga status som uppfyllda eller om det fanns behov för vidareutveckling. Resultatet visade sig vara acceptabelt för Knowit.

Krav 1.2 "Autentiseringstjänsten ska ha stöd för social inloggning" kunde inte fullt demonstreras i modulen eftersom Okta i nuläget tvingar utvecklaren att antingen hosta inloggningswidgeten själv eller en domän verifierad hos Okta för att kunna använda sig av social inloggning. En prototyp av detta uteblev på grund av tidsbrist. Istället demonstrerades ett implementationsförslag med hjälp av Oktas livewidget (se bild 3 nedan), som är en webbaserad kodsandbox för att demonstrera deras widgets anpassningsbarhet. Företaget accepterade detta och samtliga krav godkändes därmed.

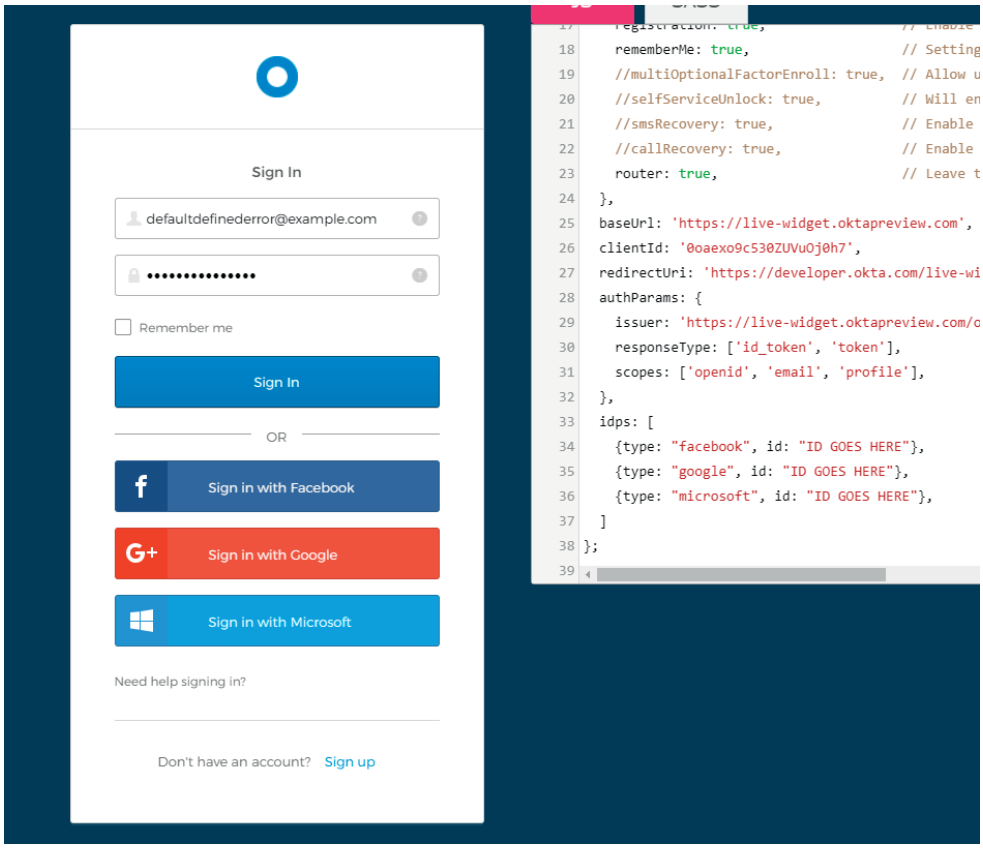


Bild 3. *Social inloggning i Oktas livewidget.*

4.6 Säkerhetsutvärdering

Resultatet av arbetets säkerhetstest redovisas i tabell 2 nedan. Varje ID presenteras i tabellen enligt följande: *G* innebär att alla testscenarion för ID-numret godkändes utan anmärkning. *Anm.* innebär att åtminstone ett testscenario godkändes med anmärkning samt att inga scenarion underkändes. *U* innebär att åtminstone ett testscenario underkändes. För överblicksbildens skull färgkodas även *G* som vit, *Anm.* som blå och *U* som orange.

Sammantaget godkändes 39 tester (varav 10 med anmärkning) medan 6 tester underkändes. De underkända testerna kommenteras nedan. För

detaljerade resultat och kommentarer för varje enskilt testscenario, se bilaga 5.

Urvalet av OWASP-tester som utfördes resulterade i 6 underkända tester. Dessa rörde sig dels om icke-konfigurerbar funktion i Okta (som röjning av redan registrerade e-postadresser i registreringsformuläret och avsaknad av komplexitetskrav på användarnamn), dels om serverkonfiguration (som användning av svaga cipher suites och TLS-versioner) och saknat skydd för cookie replay i ASP.NET.

OWASP ID	Status	OWASP ID	Status	OWASP ID	Status
WSTG-IDNT-01	Anm.	WSTG-SESS-01	Anm.	WSTG-CLNT-06	G
WSTG-IDNT-02	G	WSTG-SESS-02	Anm.	WSTG-CLNT-07	G
WSTG-IDNT-03	G	WSTG-SESS-03	G	WSTG-CLNT-08	G
WSTG-IDNT-04	U	WSTG-SESS-04	U	WSTG-CLNT-09	Anm.
WSTG-IDNT-05	U	WSTG-SESS-05	Anm.	WSTG-CLNT-10	G
WSTG-ATHN-01	G	WSTG-SESS-06	U	WSTG-CLNT-11	G
WSTG-ATHN-02	G	WSTG-SESS-07	Anm.	WSTG-CLNT-12	G
WSTG-ATHN-03	G	WSTG-SESS-08	G	WSTG-CLNT-13	Anm.
WSTG-ATHN-04	G	WSTG-SESS-09	Anm.	WSTG-CLNT-14	Anm.
WSTG-ATHN-05	G	WSTG-CRYP-01	U		
WSTG-ATHN-06	G	WSTG-CRYP-02	G		
WSTG-ATHN-07	G	WSTG-CRYP-03	G		
WSTG-ATHN-08	U	WSTG-CRYP-04	G		
WSTG-ATHN-09	G	WSTG-CLNT-01	G		
WSTG-ATHN-10	G	WSTG-CLNT-02	G		
WSTG-ATHZ-02	G	WSTG-CLNT-03	Anm.		
WSTG-ATHZ-03	G	WSTG-CLNT-04	G		
WSTG-ATHZ-04	G	WSTG-CLNT-05	G		

Tabell 2. *Tabell över resultat av säkerhetstest.*

5 Slutsats

I detta kapitel besvaras de problemformuleringar som gjordes i början av arbetet.

5.1 Om det finns någon, hur ser den nuvarande autentiseringslösningen ut?

Granskning av den ursprungliga autentiseringslösningen och samtal med företaget visade att lösningen var otillräcklig med stora brister som att kunder inte kunde registrera sig själva. Under prototyputveckling blev det uppenbart att ingen del av den ordinarie autentiseringslösningens användargränssnitt skulle bevaras vid anslutning till en extern autentiseringstjänst.

5.2 Vilka fördelar prioriteras under utvärderingen?

Företaget prioriterade låg kostnad, enkelhet att sätta upp och konfigurera samt stöd för MFA och social inloggning. En förmånlig gratisplan var enligt företaget viktigt för beslutet och var en av argumenten för att Okta valdes.

5.3 Hur ska autentiseringsalternativen utvärderas?

Utvärderingen gjordes baserad på kostnad, användarvänlighet och säkerhet. Det visade sig dock svårt att göra någon djupare bedömning av de två sistnämnda genom att bara granska dokumentation och internetforum. Det var därför av stor vikt för jämförelsen att en enklare prototyp utvecklades av båda tjänsterna eftersom det var då bristerna i användarvänlighet och säkerhet infann sig.

Med en djupare förförståelse för autentiseringsprotokoll och relaterade standarder hade en noggrannare utvärdering av säkerheten kunnat göras redan innan prototyputvecklingen.

5.4 Vilket autentiseringsalternativ lämpar sig bäst som företagets nuvarande webbapplikation?

Kostnadsskillnaden mellan Okta och Auth0 ansågs vara marginell med deras nuvarande betalningsplan. Således är både Okta och Auth0 lämpliga kandidater för företagets webbapplikation. Den avgörande faktorn blev till slut Oktas uppköp av Auth0-teamet.

Det bäst lämpade alternativet för Knowits webbapplikation var därför Okta.

5.5 Hur ska en *flexibel* autentiseringslösning implementeras?

Arbetet fastställde företagets definition på *flexibel* som användarvänlig och konfigurerbar för utvecklare och förvaltare av applikationen. Den utvecklade modulen är inte beroende av en särskild implementation av OAuth2 eller OpenID Connect. Med andra ord finns möjligheten att i framtiden byta autentiseringstjänst från Okta till en godtycklig tjänst som implementerar en API med dessa standarder.

Ytterligare flexibilitet låg i den valda autentiseringstjänstens möjlighet att konfigurera autentiseringsvillkor för användare. Oktas webbgränssnitt kan användas för att konfigurera detaljer som grupptillhörigheter, inloggningspolicies och levnadslängd på sessioner och tokens. På så vis kan applikationens beteende anpassas utan att applikationens webb-konfiguration eller källkod behöver ändras.

5.6 Hur ska en *användarvänlig* autentiseringslösning implementeras?

Att autentiseringslösningen skulle ske genom en implementation av authorization code flow eller hybrid flow innebar att användaren skulle behöva bli omdirigerad till autentiseringstjänsten för att kunna autentisera sig. Detta innebar att autentiseringstjänsten behövde ha stöd för utvecklare att skräddarsy inloggningsskärmen i enlighet med webbapplikationens utseende för att minska risken att förvirra slutanvändaren. Den valda autentiseringstjänsten Okta hade stöd för denna typ av funktionalitet.

Under arbetet gång insågs det att användarvänlighet och säkerhet ibland kunde vara varandras motsatser. En säkerhetsaspekt som att ha kortlivade access och refresh tokens kunde, förutom att vara säkrare, ha negativ påverkan på en slutanvändares upplevelse då denna kanske skulle bli tvungen att omautentisera sig efter endast kort inaktivitet i webbapplikationen.

Arbetets brist på oberoende användbarhetstester gjorde att denna fråga inte fullt kunde besvaras, i enlighet med Lauesens råd [11].

5.7 Hur ska en *säker* autentiseringslösning implementeras?

När en extern tjänst används för autentisering och rättighetstilldelning är det desto viktigare att definiera var tjänstens ansvar slutar och var den egna integrationens ansvar tar vid. Att enbart gå efter tjänstens officiella dokumentation gav många bra tips för att öka och verifiera säkerheten i dess användning, men det är ingen garanti på ett säkert system. Vid integration av dessa tjänster kan gränssnittet till en redan befintlig applikation behöva anpassas. Detta riskerar introducera osäkra lösningar som tjänstens dokumentation inte har övervägt.

Det är således viktigt att studera oberoende internetstandarder av de protokoll eller tekniker som tjänsten använder sig av. På så vis kan utvecklare med större säkerhet garantera att ens användning av tjänsten vidtagit de säkerhetsåtgärder som rekommenderas. Även om det görs ett gediget säkerhetsarbete hos stora aktörer som Okta, Auth0 och Keycloak, är det inte tillräckligt att blint förlita sig på att dessa tjänster ska täcka alla säkerhetsbehov. Ansvaret ligger hos utvecklaren som använder tjänsterna.

För att implementera en säker autentiseringslösning är det inte fullgott att endast säkerhetstesta de områden som direkt ansluter till autentiseringen. I kringliggande konfiguration finns även komponenter som bidrar till säkerhet. Exempel på dessa är användning av TLS och försäkran om att synliga variabler på användarsidan inte avslöjar information om vilken serverteknologi som används.

Även om testkategorier som ansågs ligga nära autentisering valdes, innefattar kategorierna områden som inte strikt behandlar autentiseringen i sig utan även kringliggande konfiguration. En isolerad modul kan göras säker, men om det finns svaga länkar i resten av systemet komprometteras en applikations säkerhet likväl. Således är det viktigt att tänka på säkerhetsarbete som att införa flera lager av säkerhet, och inte försöka hitta den definitiva lösningen som ska skydda systemet från allt. Som introduktionen i [12] så vackert förtäljer: *"There is no silver bullet"*.

6 Diskussion

I detta kapitel diskuteras etiska aspekter som rör arbetet och dess resultat. Därefter föreslås framtida utvecklingsmöjligheter.

6.1 Reflektion över etiska aspekter

I dagens samhälle sker mer och mer kommunikation över internet och känslig data finns allt oftare tillgängligt via internet. Det är nu mer än någonsin viktigt att säkerställa vem det är som har tillgång till känslig data. Enligt webbtidningen *Cyber Security Ventures* beräknas hacking kosta cirka sex biljoner amerikanska dollar år 2021 [23]. Det är därför i samhällets goda intresse att försöka förhindra detta. Genom säkerhetsorganisationer som OWASPs arbete och design samt utförande av test liksom de i detta arbete kan många av de vanliga säkerhetsbrister i webbapplikationer belysas och elimineras och således kriminell aktivitet som hacking försvåras.

Behovet av att kunna autentisera sig på nya sätt som med t.ex MFA eller social inloggning ökar med tiden eftersom detta anses både säkert och, i social inloggnings fall, användarvänligt. Autentiseringstjänster som Okta och Auth0 är kända exempel på hur sådan funktionalitet kan implementeras där fler kända företag börjar använda dessa som autentiseringstjänst. Det är därför av stor vikt att granska både tjänsternas säkerhet i sig och en färdig implementation av en av dessa.

6.2 Framtida utvecklingsmöjligheter

Framtida utvecklingsmöjligheter kan innefatta att ytterligare autentiserings-tjänster undersöks och inte bara de två som undersöktes i detta arbete. Om företaget hade varit intresserade av en egenutvecklad autentiseringstjänst i form av en molnlösning hade fördelar och nackdelar där också kunnat undersökas.

Analys av en liknande implementation för ASP.NET Core hade varit av framtidsintresse, då EPiServer planerar stödja detta ramverk [24]. Skillnaderna i uppsättning bör ej vara stora eller särskild många, men att kartlägga fallgropar under implementering kan hjälpa framtida projekt vara

mer observanta på vanliga misstag vid utveckling och säkerhets-konfiguration som lätt kan förbises.

En annan intressant analys hade varit av en autentiseringslösning som strikt använder Oktas Authentication API för att hantera autentisering och sessioner. Att utforska vanliga misstag och fallgropar i en proprietär modul som använder en sådan lösning hade kunnat belysa säkerhetsbrister i ett sådant system. Samtidigt hade Oktas officiella guider och dokumentation för användning av API:n kunnat granskas ur ett säkerhetsperspektiv.

Ytterligare ett intressant studieområde hade varit en applikation som enbart använder sig av OpenID Connect med frontendkod. En sådan applikation ställer helt andra krav på implementation för att kunna bedömas vara säker.

7 Källförteckning

[1] EPiServer, *What is a CMS?,* feb 2018. Hämtad: apr 2021. [Online] Tillgänglig: https://world.episerver.com/documentation/developer-guides/CMS/learning-path/what-is-a-cms/

[2] D. Hardt, *RFC 6749 - The OAuth 2.0 Authorization Framework*, okt 2012. Hämtad: mar 2021. [Online]. Tillgänglig: https://tools.ietf.org/html/rfc6749

[3] M. Jones, J. Bradley, N. Sakimura, *RFC 7519 - JSON Web Token*, maj 2015. Hämtad: mar 2021. [Online]. Tillgänglig: https://tools.ietf.org/html/rfc7519

[4] N. Sakimura, J. Bradley, B. Medeiros, *Final: OpenID Connect Core 1.0 incorporating errata set 1,* nov 2014. Hämtad: apr 2021. [Online] Tillgänglig: https://openid.net/specs/openid-connect-core-1_0.html

[5] M. Jones, J. Hildebrand, *JSON Web Encryption*, maj 2015. Hämtad: apr 2021. [Online] Tillgänglig: https://tools.ietf.org/html/rfc7516

[6] The OWASP Foundation, apr 2021. Hämtad: apr 2021.[Online] Tillgänglig: https://owasp.org/

[7] The OWASP Foundation, *OWASP ZAP Zed Attack Proxy,* maj 2021. Hämtad: maj 2021.[Online] Tillgänglig: https://owasp.org/www-project-zap/

[8] PortSwigger, *Features,* maj 2021. Hämtad: maj 2021.[Online] Tillgänglig: https://portswigger.net/burp/enterprise/features

[9] Nmap, *Introduction*. [Online] Hämtad: maj 2021. Tillgänglig: https://nmap.org/

[10] Wireshark, *About Wireshark,* maj 2021.[Online] Hämtad: maj 2021. Tillgänglig: https://www.wireshark.org/

[11] S. Lauesen, *Software Requirements,* Addison Wesley, 2002, sid. 253.

[12] The OWASP Foundation, *WSTG - v4.2,* dec 2020. Hämtad: apr 2021. [Online] Tillgänglig: https://owasp.org/www-project-web-security-testing-guide/v42/

[13] R. Miller, *Okta acquires cloud identity startup Auth0 for $6.5B, TechCrunch, mar 2021,* Hämtad: maj 2021. [Online] Tillgänglig: https://techcrunch.com/2021/03/03/okta-acquires-cloud-identity-startup-auth0-for-6-5b/

[14] Ad Fontes Media, *TechCrunch Bias and Reliability Overview,* Hämtad: maj 2021. [Online] Tillgänglig: https://www.adfontesmedia.com/techcrunch-bias-and-reliability/

[15] Okta, *Customers,* Hämtad: maj 2021. [Online] Tillgänglig: https://www.okta.com/customers/

[16] Auth0, *Customers,* Hämtad: maj 2021. [Online] Tillgänglig: https://auth0.com/customers/

[17] Okta, *OpenID Connect & OAuth 2.0 API*, Hämtad: maj 2021. [Online] Tillgänglig: https://developer.okta.com/docs/reference/api/oidc/

[18] Auth0, *OAuth 2.0 Authorization Framework,* Hämtad: maj 2021. [Online] Tillgänglig: https://auth0.com/docs/protocols/protocol-oauth2

[19] OpenID Connect, *OpenID Certification,* . Hämtad: maj 2021. [Online] Tillgänglig: https://openid.net/certification/

[20] F. Lardinois, *Okta launches a new free developer plan,* TechCrunch, apr 2021. Hämtad: maj 2021. [Online] Tillgänglig: https://techcrunch.com/2021/04/06/okta-launches-a-new-free-developer-plan/

[21] OWIN, *OWIN - Open Web Interface for .NET,* Hämtad: maj 2021. [Online] Tillgänglig: http://owin.org/

[22] P. Ignatius, *Säkerhetstestning av webbapplikationer och CMS plattformen EPiServer,* Högskolan Dalarna, maj 2011. Hämtad: maj 2021. Tillgänglig: https://www.diva-portal.org/smash/get/diva2:519089/FULLTEXT01.pdf

[23] S. Morgan, *Cybercrime To Cost The World $10.5 Trillion Annually By 2025,* Cybersecurity Ventures, nov 2020. Hämtad: apr 2021. Tillgänglig: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016

[24] M. Ottosen, *ASP.NET Core beta program,* dec 2019. Hämtad: maj 2021. Tillgänglig: https://world.episerver.com/blogs/martin-ottosen/dates/2019/12/asp-net-core-beta-program/

8 Appendix

Bilaga 1. Intervjudokument

Måndag 2021-03-29

Ett öppet samtal med handledarna på Knowit ledde till följande, preliminära krav.

AUTENTISERINGSTJÄNST

1) Det ska vara lätt att **förvalta** och **konfigurera** autentiseringstjänsten.
    a) *"Den valda autentiseringstjänsten ska efter företagskontakt bedömning vara lättnavigerad i sitt webbgränssnitt."*
    b) "Den valda autentiseringstjänsten ska vara flexibel och således gå att använda i knowits framtida projekt"
2) Autentiseringstjänsten ska ha stöd för social inloggning
    a) identity providers såsom Facebook, Google.
3) Autentiseringstjänsten ska ha stöd för multifaktoriell autentisering.

MODUL

1. Modulen synkroniserar roller från autentiseringstjänsten med rättigheter i EPiServer.
2. Koden för modulen ska versionshanteras via bitbucket.
3. Modulen ska överta ansvaret för autentisering och rollhantering enligt gränssnittet "*IIdentityService*".
4. Modulen ska logga alla inloggningsförsök och spara dessa till en fil.
5. Modulen ska ge en varning i loggfilen om en roll i autentiseringstjänsten inte stämmer överens med en roll i epi.
6. Modulen ska vara utredd enligt utvalda OWASP säkerhetstest

DOKUMENTATION

1. Koden ska vara, enligt Knowit, väldokumenterad.
2. En installation- och konfigurationsguide ska skrivas för autentiseringslösningen.

Tisdag 2021-04-06

En semi-strukturerad intervju fördes med handledarna på företaget.

1. **Fråga:** Vad i klassen IdentityService behöver bevaras?
   Svar: *Den kan försvinna och bytas ut mot en egen lösning!*

2. **Fråga:** Vad i klassen IdentityService behöver anpassas?
   Svar: Se ovan.

3. **Fråga:** Finns det övriga klasser som ni känner till som vi ka behöva ändra i?

Svar: *Nej, det ska vara isolerat till de features relaterade till identitet som vi pratat om hittills.*

4. **Fråga:** Kommer det finnas krav på lösenord. Behöver vi ta ställning till dessa just nu?
   Svar: *Det hanteras senare av folk som ska konfigurera. Krävs bara att det ska gå att konfigurera hos tjänsten (ey, det gör det!)*

5. **Fråga:** Kommer det behövas en rolldefiniton redo?
   Svar: *Utgå från Epis rolldefinition + icke-redaktörer (kunder!)*

6. **Fråga:** Kommer det behövas en felhantering om en roll, tilldelad i autentiseringstjänsten inte stämmer överens med en roll i epi?
   Svar: *Applikationen ska inte avbryta exekvering, men logga gärna om det sker en rollmissmatchning.*

Tisdag 2021-04-13

Ett öppet samtal fördes med handledarna på företaget kring problemställningen nedan.

> *I nuläget används access tokens från Okta/Auth0 (eg. id-token från auth0, access tog lite mer uppsättning som vi inte fått rätt på ännu), som synkar info till en ASP.NET Identity. Denna sparas - som ni hundra säkert känner till - som en cookie i läsaren, som används för autentisering.*
>
> *I nuläget tvingas användaren till en omautentisering för att uppdatera cookien, t.ex. om ändringar har skett i rolltillhörigheter, byte av namn eller lösenord eller om användaren avaktiverats i applikationen.*
>
> *En lösning är att med jämna mellanrum (säg, access tokens livslängd), låta backenden uppdatera Identityn och skapa en ny cookie så länge cookiens sparade refresh token är giltig. Detta funkar fint, men kanske är onödigt tungt eller kanske tillochmed medför andra problem.*
> *Ett uppenbart problem är ju att en rollförändring/avaktivering av ett konto inte är omedelbart, vilket gör att en användare har ett tidsfönster där den kan härja fritt.*

Samtalet ledde till den föreslagna lösningen accepterades. Autentiseringskakor ansågs vara tillräckligt säkrade via TLS.

Måndag 2021-04-19

En semi-strukturerad intervju fördes med handledarna på företaget.

1. **Fråga:** Vad för slags tester förväntar ni er att autentiseringen ska kunna klara av?
   Vi fokuserar nu mycket på säkerhetstester, och tar gärna emot förslag på vad vi kan göra i KEX för att testa funktionalitet för modulen. Vad är aktuellt i detta skede, hämtning av

användarinfo, varukorg och liknande?
Svar: *Registrera sig, skapa konto, förhöjda privilegier, använda roller för Visitor Groups, lägga till i varukorgen, hämta användarinfo.*

2. **Fråga:** Hur ska vår modul sköta eventuell felrapportering och loggning?
   Svar: *Använd EPiServers ILogger-interface.*

3. **Fråga:** Hur löser vi GUID? Autentiseringstjänsterna förser inte varje användare med ett id som är 32-tecken långt och hexadecimalt. Skulle vi kunna använda en debug-lösning med MD5-hash i nuläget, trots att det inte är kollisionsresistent?
   Svar: *Definiera detta som ett known issue. Lösningar är fullgod för detta utvecklingsstadie och kan lätt bytas ut senare.*

4. **Fråga:** Har ni hunnit titta genom jämförelsedokumentet och bestämt er för en tjänst ännu?
   Svar: *Ja, Okta känns rimligt då de samtidigt tar upp Auth0-teamet i sitt företag.*

Måndag 2021-05-03

1. **Fråga**: Hur ska rolldefinitionen fungera? Ska det sättas i EPi eller i Okta? Ska roller som sätts i EPi skrivas över när roller importeras från Okta?
   Svar: *Rollsynkning från Okta kan utebli. Istället kan rollerna hanteras direkt i EPiServer.*

2. **Fråga:** Får vi ha med säkerhetstesten med avseende på NDA. Hur mycket om systemet får vi beskriva? Ska vi skriva att det vi testar är knowit/episerver/kex/aspnet?
   Svar: *Systemet kan benämnas som webbapplikationen. Säkerhetsbrister i EPiServer eller ASP.NET-ramverket kan ordas om fritt. Genomgång av testresultat kan behöva göras för att sanera den från punkter som kan användas för riktade cyberattacker.*

Bilaga 2. Kravspecifikation

Autentiseringstjänstskrav

~~1.1. Det ska vara lätt att förvalta och konfigurera autentiseringstjänsten.~~

1.2. Autentiseringstjänsten ska ha stöd för social inloggning

1.3. Autentiseringstjänsten ska ha stöd för multifaktoriell autentisering.

1.4. Autentiseringstjänsten ska kunna ställa krav på lösenordskomplexitet.

~~1.5. Registrering av ett nytt konto följer de komplexitetskrav som ställs på lösenord i autentiseringstjänsten.~~

1.6. Autentiseringstjänsten ska endast tillåta ändring till lösenord som skiljer sig från användarens 8 senaste lösenord.

1.7. Autentisteringstjänsten ska endast tillåta ändring av lösenord som är minst XXX timmar gamla.

1.8. Autentiseringstjänsten ska kunna forcera användaren att byta lösenord efter av administratörer godtyckligt angiven tid. (s.k. stöd för password expiration)

1.9 Tjänsten ska kunna tillåta användaren att manipulera vad en token innehåller direkt i webbgränssnittet.

1.10. Den valda autentiseringstjänsten ska enligt företagskontakts bedömning vara lättnavigerad i sitt webbgränssnitt.

1.11. Den valda autentiseringstjänsten ska enligt företagskontakts bedömning vara enkelt konfigurerbar i företagets liknande ASP.NET-projekt.

Modulkrav

~~2.1. Modulen synkroniserar användare med roller från autentiseringstjänsten med EPiServers databas.~~

2.2. Koden för modulen ska versionshanteras via bitbucket.

2.3. Den utvecklade modulen ska helt ersätta utvecklingmiljöns nuvarande implementation av IIdentityService.

2.4. Modulens standarddefinition för roller ska utgå från EPiServers rolldefinition, med tillägget av en Customer-roll.

~~2.5. Modulen ska logga autentiseringsförsök innehållandes rollanspråk för roller som ej är definierade i modulen.~~

2.6. Modulen innehåller en funktion som kan hämta den inloggade användarens GUID.

2.7 Modulen kan skapa en anonym cookie för icke-autentiserade användare.

2.8. Autentiseringscookien uppdateras med ett tidsintervall som bestäms av sin access tokens levnadstid.

2.9. Autentiseringscookien kan uppdateras så länge dess refresh token är giltig.

2.10 Med hjälp av modulens GUID-funktion, ska applikationen kunna hämta en användares varukorg.

~~2.11 Användares roller som synkats till EPi ska kunna användas i funktionen Visitor Groups~~

2.12 När användaren loggar ut från applikationen loggas användaren även ut från autentiseringstjänsten.

2.13 När användaren loggar ut rensas autentiseringscookie och sessionscookie bort från användarens webbläsare

2.14 Användarens roller kan styras genom EPiServers gränssnitt.

Dokumentationskrav

3.1. Alla funktioner i modulen ska dokumenteras med sammanfattning (summary), parameterbeskrivning (param) och eventuella anmärkningar (remarks).

3.2. En installation- och konfigurationsguide ska skrivas för autentiseringslösningen.

3.3. Alla properties ska vara namngivna på ett sådant vis att namnet beskriver propertyns ansvar.

Säkerhetskrav

1. Applikationen har säkerhetsutvärderats med test baserade på Identity Management Testing i OWASP Testing Guide v4.2.

4.2. Applikationen har säkerhetsutvärderats med test baserade på Authentication Testing i OWASP Testing Guide v4.2.

4.3. Applikationen har säkerhetsutvärderats med test baserade på Authorization Testing i OWASP Testing Guide v4.2.

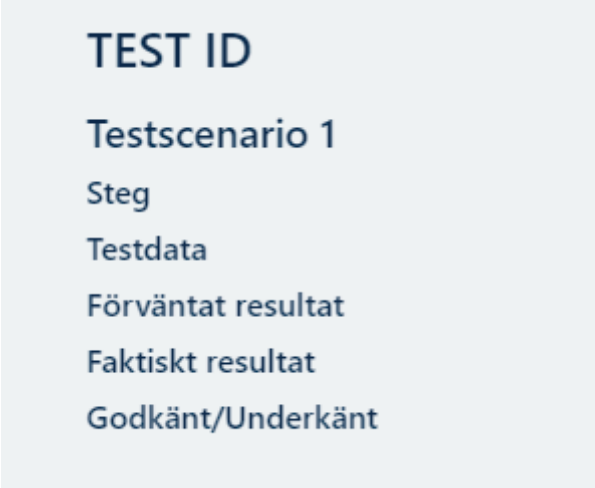
4.4. Applikationen har säkerhetsutvärderats med test baserade på Session Management Testing i OWASP Testing Guide v4.2.

4.5. Applikationen har säkerhetsutvärderats med test baserade på Cryptography i OWASP Testing Guide v4.2

4.6. Applikationen har säkerhetsutvärderats med test baserade på Client Side Testing i OWASP Testing Guide v4.2.

Projektkrav

5.1 Samtliga säkerhetstest ska vara specificerade enligt rubrikerna nedan:



5.2 OWASP-tester dokumenteras i ett separat dokument.

5.3 Övriga tester (d.v.s. inte OWASP-tester) dokumenteras i motsvarande kravs Trellokort.

5.4 Om ett krav verifieras av ett redan etablerat test, anges detta i kravets Trellokort.

5.5 Samtliga tester ska sammanställas i ett dokument som sedan läggs till som bilaga i examensrapporten.

5.6 Samtliga krav ska finnas i projektets trello.

Bilaga 3. Jämförelsedokument

AUTH0

Kostnad

9 791 SEK i månaden för upp till 50 000 MAUs

Autentisering för en applikation med tjänstens Developer-prisplan. Gratisplan finns även dock är denna begränsad till 7000 MAUs och 2 identity providers.

MAU (antal)	Kostnad ($) Månadsvis
1000	$23(gratisplan 0$)
2500	$57(gratisplan 0$)
5000	$114(gratisplan 0$)
7000	$160(gratisplan 0$)
10000	$228
20000	$455
50000	$1138

Identity Providers

51 sociala idp där man får ha ett obegränsat antal samtidigt.
Multifaktoriell autentisering
7 olika mfa-lösningar. Vid Enterprice(Kontakta auth0 för pris) kan man använda auth0s egna mobilauthenticator som heter auth0 Guardian.

SSO-stöd

ja

Skalbarhet

Kontakt med auth0 för evaluering av pris vid över 50 000 MAUs.

Användbarhet

Väldigt detaljerade guider som steg för steg visar integrering och uppstart av tjänsten. Exempelprojekt för ett tiotal olika serverramverk går att ladda ner för att testa tjänsten på t.ex ASP.NET, java eller PHP.

Konfigurera en identity provider

Från administratörens dashboard klicka på **add social connctions > create connection > önskad idp > continue >** client ID, client secret och rättigheter som applikationen har för idp-kontot > **create > Klicka på önskad applikation för att lägga till idp för denna. (6 klick)**

Snabbt och enkelt från dashboarden. Lite oklart om det sparas efter man lagt till IDPn i en applikation men vid extra kontroll kan man bekräfta att den är sparad.

Konfigurera multifaktoriell autentisering

One time password: Från administratörens dashboard navigerar man till > **Setup Multi-factor Auth > för vald faktor tryck på "enable" > definera vilken av tre polices som ska användas > save (4 klick)**

Snabbt och enkelt från dashboarden. Gör precis vad man vill och minimalt antal klick. Extra steg om man vill använda mobilnummersautentisering eller auth0 andra autentiseringslösningar.

Skapa en användargrupp

Från administratörens dashboard klicka på **user management > roles > create role >** fyll i namn och beskrivning > **create (4 klick)**

Man måste ha skapat en API i auth0 och skapat permissions för att så dessa ska gå att välja i detta steg men detta förklarar auth0 väldigt tydligt.

Konfigurera en grupps claims

Måste göras med kod. Från administratörens dashboard tryck på > **auth pipeline > rules > create rule > empty rule >** skriv in kod där du ändrar vad som ska skickas och sparas i access- eller id-token, finns hjälp online och på auth0-docs om detta > **save changes (5 klick)**

Lite synd att detta inte verkar gå fixa utan kod just nu. Lite omständigt.

Aktivera CORS / Trusted Origins

Från administratörens dashboard klicka på **applications > applications > klicka på din applikation(...) > settings >** på sidans nedre del, fyll i adress i fältet för trusted origins (Allowed Origins (CORS)) > **save changes (5 klick)**

Inte helt uppenbart var detta låg men gick att gissa sig till det. Hade kanske varit bättre med underkategorier istället för en väldigt lång sida där man behöver scrolla långt men fortfarande hanterbart.

Enkelhet att integrera och konfigurera

Auth0 erbjuder flera guider och kodexempel för flera olika av ramverk och kodspråk, däribland specifikt för ASP.NET Core och MVC. En grundläggande integration och konfiguration bedöms vara enkel att utföra.

Säkerhet

Har OAuth2-stöd, vilket kan göras säkert
Revokering av access token inte möjligt. Dessa är dock kortlivade.

OKTA

Kostnad

Autentisering för en applikation med tjänstens Developer-prisplan.

MAU (antal)	Kostnad ($) Månadsvis
1000	$0
2500	$50
5000	$100
7000	N/A
10000	$200
20000	$400
50000	$1000

Identity Providers

Okta implementerar stöd för social autentisering via Google, Facebook, LinkedIn, Microsoft, Apple, OpenID Connect IdP och SAML 2.0 IdP.

Multifaktoriell autentisering

Prisplanen **Developer** har grundläggande stöd för MFA med en egen lösning Okta Verify. För komplett stöd med tredjepartslösningar (Google Auth m.m) krävs en plan för över 50000 MAUs (One App- eller Enterprise-planen).

Användbarhet

Lägga till eller konfigurera en identity provider

Från administratörens dashboard navigerar man till **Security > Identity Providers > Add Identity Provider**. **Välj** önskad identity provider från listan. Klientinformation och secret matas in. Bekräfta inlägget med **Add Identity Provider**. Att konfigurera en provider har lika många klick. (5 klick).

När man väl hittat till inställningen är det väldigt lättarbetat. Dock är det inte självklart att man ska leta under fliken **Security** till att börja med.

Lägg till multifaktoriell autentisering

Från administratörens dashboard navigerar man till **Security > Multifactor. Välj** önskad faktor i listan. **Klicka** på Active/Inactive-dropdownlistan. **Klicka** Active. (5 klick)

Upplevs enkelt och inställningen känns lämpligt placerad i gränssnittet. Enkel att navigera till.

Konfigurera multifaktoriell autentisering

Från administratörens dashboard navigerar man till **Security > Multifactor > Factor Enrollment. Klicka** Edit. För vald faktor, **tryck** på dropdownmenyn. **Välj** alternativ**.** (6 klick)

Upplevs enkelt. Tydligt hur man kan lägga till andra policies som kan gälla för olika användargrupper.

Skapa en användargrupp

Från administratörens dashboard navigerar man till **Directory > Groups > Add Group.** Namn och beskrivning skrivs in och sedan klickas **Add Group** för att bekräfta inlägget. (4 klick)

Relativt lätt att navigera till. Begreppet **Directory** upplevdes inte helt självklart.

Konfigurera en grupps claims

Man kan lägga till grupper som custom claims i token (både ID och Access). Dessa kan även filtreras enligt enklare regler i Oktas gränssnitt. Detta var relativt enkelt via inställningarna under **Security > API > Authorization Servers**. Önskad auktoriseringsserver **väljs**. Navigera till **Claims.**

Här kan claims läggas till - antingen att grupptillhörighet läggs till (med eventuell filtrering) eller skrivs uttryck som utgår från användarens attribut (användarnamn, epostadress och så vidare).

Aktivera CORS / Trusted Origins

Från administratörens dashboard navigerar man till **Security > API > Trusted Origins > Add Origin.** Där skrivs domännamn in, CORS-alternativet väljs och sedan sparar man inlägget. (6 klick)

Kändes relativt enkelt. Hade uppskattat en genväg som nämner CORS, men när man väl är medveten om var man hittar det är det en rimlig plats för det.
Enkelhet att integrera och konfigurera
Okta erbjuder flera guider och kodexempel för flera olika av ramverk och kodspråk, däribland specifikt för ASP.NET Core och MVC. En grundläggande integration och konfiguration bedöms vara enkel att utföra.

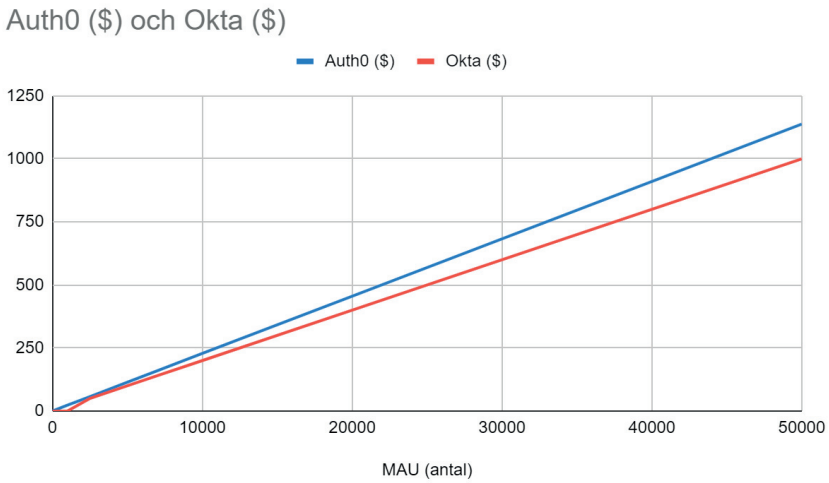
Säkerhet

Okta har stöd för OAuth2-protokollet med OpenID Connect.
Revokering av access och refresh tokens är möjligt. Vid kompromitterad access token kan den direkt inaktiveras. Att invalidera access tokens är egentligen inte nödvändigt ur ett säkerhetsperspektiv, om dessa tokens har kort livslängd.

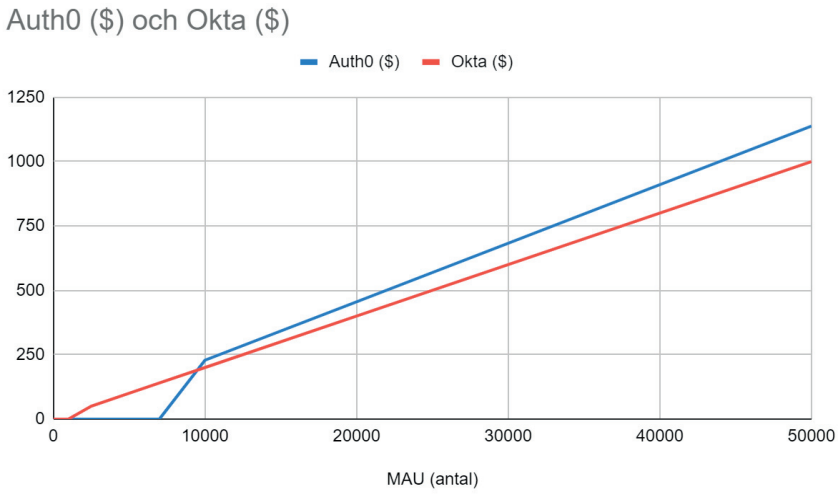
Slutsats

Kostnad

Lösningarna har likartad kostnad. Auth0 är något dyrare i sin Developer-plan men inkluderar komplett MFA. I Okta är MFA ett tillköp. Det är i nuläget oklart hur tjänsterna ökar i pris när man överskrider 50000 MAUs men enligt Per och Mathias på Knowit är detta något vi inte behöver ta hänsyn till i nuläget.



Om auth0s gratisplan skulle vara av intresse ser kostnadsjämförelsen ut så här:



Identity Providers
Auth0 har ett större utbud av identity providers. Dock har Okta stöd för de mest populära (Google, Facebook, LinkedIn, Microsoft).

Multifaktoriell autentisering
Auth0s Developer-plan inkluderar komplett MFA. I Okta är MFA ett tillköp - Developerplanen har grundläggande MFA med Oktas egna system Okta Verify (inte bekant hos slutanvändare).

Användbarhet
Lägga till eller konfigurera en identity provider
Auth0 har ett tydligare användarflöde när man konfigurerar identity providers. Oktas inställningar för IdP kräver färre klick, men är mindre intuitiv att navigera till. Båda är smidiga att konfigurera i webbgränssnittet.

Lägg till multifaktoriell autentisering
Båda är välplacerade och smidiga att konfigurera i webbgränssnittet.

Konfigurera multifaktoriell autentisering
Båda är välplacerade och smidiga att konfigurera i webbgränssnittet.

Skapa en användargrupp
Båda är enkla att navigera till, även om Oktas flik “Directory” kunde varit tydligare namngiven. Okta kräver inte att man skapar API eller applikationen innan man skapar grupper - dessa kan länkas ihop i efterhand.

Konfigurera en grupps claims
Båda kräver en del klick för att navigeras till. Auth0s breadcrumb känns mer genomtänkt. Oktas custom claims kan endast läggas till i gruppform, och får sedan mappas till rättigheter i applikationen. Detta är ingen nackdel, då en RBAC-approach ändå är att föredra. Claims måste skrivas i kod i auth0 vilket gör det lite mer svåråtkomligt om man inte har erfarenhet av detta. Dock bra att man kan skriva kod direkt i webbgränssnittet och att det finns mycket docs för detta.

Aktivera CORS / Trusted Origins
CORS i Okta var enklare att navigera till än i auth0 eftersom detta i auth0 låg längst ner i applikationens settings.

Enkelhet att integrera och konfigurera
Då båda tjänsterna har stöd för OAuth2 med OpenID Connect bedöms de vara likvärdiga

Säkerhet
Vid första anblick är säkerheten likvärdig. Det är upp till klienten att validera tokens korrekt vid användning av OAuth2 / OpenID Connect. Auth0 har, till skillnad från okta, inget stöd för revokering av accesstokens vilket är en säkerhetsbrist ifall dessa skulle bli kapade. Vid client credential flow kan säkerhetsbrister uppkomma i båda tjänster eftersom en token kan bli exponerad i frontenden. Detta är dock något som både auth0 och okta avråder från om autentiseringen går att lösa genom t.ex authorization code flow vilket båda räknar som det bästa sättet att hantera autentisering och auktorisering.

Med detta som bakgrund antas båda tjänsterna vara lika säkra så länge man använder authorization code flow. Om inte faller ändå ansvaret på den egna webbapplikationen att inte hålla client id osäkert lagrat. Vidare analys av eventuella säkerhetsbrister i den valda tjänsten görs efter en fullständig implementation senare i arbetet.

Sammanfattning

Efter att ha gått igenom samtliga punkter enligt kravspecifikation skapad efter diskussion med Knowit kan vi säga att både Okta och Auth0 är att rekommendera eftersom båda hade tillfredsställt de behov Knowit har. En implementation av båda i Episerver anses fullt möjligt.

Bilaga 4. Testspecifikation baserad på OWASP Web Security Guide

WSTG-IDNT-01

Testscenario 1: Dokumentation av roller som används i applikationen

Förväntat utfall
Autentiseringsmodulen innehåller standarddefinierade roller från EPiServer, med tillägget *Customers*.

Applikationen innehåller roller definierade av Knowit.

Testdata
Användarnamn: *administrator@example.com*
Lösenord: *flexauth2121*

Utförande
1. Notera vilka användargrupper som finns definierade under *https://localhost/episerver*
2. Notera vilka användargrupper som finns definierade i autentiseringstjänsten (*Okta*).

Testscenario 2: Ändra användarroller genom manipulation av sessionstoken

Förväntat utfall
Användaren kan inte manipulera autentiseringscookien på ett rimligt vis, då den är krypterad av backenden.

Testdata
N/A

Utförande
1. Logga in i webbapplikation som valfri användare.
2. Öppna inspectorn och försök ändra cookiens värde så att du erhåller en annan roll.

Testscenario 3: Se över granulariteten i rollernas rättigheter.

Förväntat utfall
Då rollernas definition fortfarande är i utvecklingsstadie, kan de endast kommenteras utifrån rättigheterna i EPiServers standardroller.

Testdata
N/A

Utförande
1. Navigera till autentiseringstjänstens grupp- och rolldefinition.
2. Notera vilka grupper som finns definierade här.
3. Navigera till EPiServers användargränssnitt för grupper/roller.
4. Notera vilka grupper som finns definierade här.
5. Kommentera upplösningen i de funna rollerna.

WSTG-IDNT-02

Testscenario 1: Registrering enligt säkerhetskrav

Förväntat utfall
Registreringen lyckas.

Testdata
Valfritt användarnamn som inte redan finns som registrerad användare.
Lösenord: *flexauth2121*

Utförande
1. Öppna ett nytt inkognitofönster i valfri webbläsare.
2. Navigera till *https://localhost/account*
3. Tryck Sign up
4. Försök registrera kontot med testdatan.

Testscenario 2: Registrering som bryter säkerhetskrav
Förväntat utfall
Registreringen misslyckas.

Testdata
Valfritt användarnamn som inte redan finns som registrerad användare.

Utförande
1. Öppna ett nytt inkognitofönster i valfri webbläsare.
2. Navigera till *https://localhost/account*
3. Tryck Sign up
4. Försök registrera kontot med ett lösenord som är färre än 8 bokstäver.
5. Försök registrera kontot med ett lösenord som består av del av användarnamnet.
6. Försök registrera kontot med lösenordet “password123”.
7. Verifiera i Oktas användarkatalog att ingen ny användare existerar.

Testscenario 3: Registrering av redan registrerad e-postadress
Förväntat utfall
Registreringen misslyckas.

Testdata
Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Utförande
1. Öppna ett nytt inkognitofönster i valfri webbläsare.
2. Navigera till *https://localhost/account*
3. Tryck Sign up
4. Försök registrera kontot.

WSTG-IDNT-03
Testscenario 1: Åtkomst till administratörsfunktion i Okta som kund
Förväntat utfall
Användaren kommer inte åt panelen i Oktas slutanvändargränssnitt. Användaren släpps inte in panelen via direktlänk.

Testdata

Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Utförande

1. Öppna ett nytt inkognitofönster i valfri webbläsare.
2. Navigera till *https://dev-6673066.okta.com/*
3. Logga in med testdatan.
4. Försök komma åt administratörspanelen via webbgränssnittet.
5. Navigera till *https://dev-6673066-admin.okta.com/*

Testscenario 2: Åtkomst till administratörsfunktion i Okta som administratör

Förväntat utfall
Användaren kommer åt panelen och kan skapa användare.

Testdata
Användarkonto: BORTTAGET
Lösenord: BORTTAGET

Utförande

1. Öppna ett nytt inkognitofönster i valfri webbläsare.
2. Navigera till *https://dev-6673066.okta.com/*
3. Logga in med testdatan.
4. Försök komma åt administratörspanelen via webbgränssnittet.
5. Navigera till *https://dev-6673066-admin.okta.com/*
6. Lägg till en användare under *Directory > People > Add Person*

WSTG-IDNT-04

Testscenario 1: Uppräkning av registrerade användare via inloggningsformulär

Förväntat utfall
Ingen skillnad mellan responserna av inloggningsförsöken kan urskiljas. Attackeraren kan inte bekräfta om ett konto finns registrerat.

Testdata
Användarkonto 1: *defaultdefined@example.com*
Användarkonto 2: *notreg@example.com*
Lösenord: *wrongpassword2121*

Utförande

1. Öppna ett nytt inkognitofönster i Chrome.
2. Navigera till *https://localhost/account*
3. Öppna upp inspectorn, öppna Network-fliken.
4. Ange konto 1 som användarnamn, skriv in lösenordet.
5. Försök logga in.
6. Notera responsen i inloggningsformuläret.
7. Notera responsen i Network-fliken i inspectorn.
8. Ange konto 2 som användarnamn, skriv in lösenordet.
9. Upprepa steg 6-7.
10. Kontrollera att responsen inte skiljer sig mellan konto 1 och konto 2.

Testscenario 2: Uppräkning av registrerade användare via registreringsformulär

Förväntat utfall
Attackeraren kan inte bekräfta om ett konto finns registrerat.

Testdata
Användarkonto 1: *defaultdefined@example.com*
Lösenordet: *wrongpassword2121*

Utförande
1. Öppna ett nytt inkognitofönster i Chrome.
2. Navigera till *https://localhost/account*
3. Tryck *Sign Up*
4. Ange konto 1 som användarnamn, skriv in lösenordet, valfritt för- och efternamn.
5. Försök registrera dig.
6. Kontrollera att registreringsformuläret inte avslöjar att kontot existerar.

WSTG-IDNT-05

Testscenario 1: Granskning av stöd för säkerhetskrav på användarnamn

Förväntat utfall
Autentiseringstjänsten har stöd för att ställa krav på användarnamn och deras struktur.

Testdata
N/A

Utförande
Undersök Oktas dokumentation och webbgränssnitt efter funktion att tillsätta krav på användarnamn.

WSTG-ATHN-01

Testscenario 1: Känslig data är okrypterad.

Förväntat utfall
Samtlig dataöverföring sker via TLS och är följaktligen krypterad.

Testdata
Användarnamn: *customer@example.com*
Lösenord: *flexauth2121*

Utförande
1. Starta programmet Wireshark.
2. Navigera till *https://localhost/account* efter att ha börjat lyssna på samtlig trafik från och till denna domän.
3. Kontrollera samtliga paket i Wireshark efter känslig data som t.ex. lösenord eller användarnamn i klartext.

WSTG-ATHN-02

Testscenario 1: Defaultvärden på nyskapade konton

Förväntat utfall
Det finns inget defaultvärde för nyskapade konto i Okta eftersom applikationen tvingar en att använda ens email och ett nytt lösenord.

Utförande

1. Öppna Google Chrome surfa till *https://localhost/account*
2. Klicka på “Registrera dig”
3. Försök skapa ett konto utan att fylla i lösenord eller email.

WSTG-ATHN-03

Testscenario 1: Utlåsning vid upprepade felaktiga lösenordsförsök

Förväntat utfall

Man blir utelåst efter det antal felaktiga lösenord som man specificerat i autentiseringstjänsten. Okta tillåter en IP-adress att skriva fel lösenord 10 gånger sen blockeras användaren.

Utförande

1. Öppna en inkognitoflik i Google Chrome.
2. Skriv in *customer@example.com* som användarnamn/email och skriv in fel lösenord tre gånger.
3. Försök nu skriva in rätt lösenord och bekräfta om du blivit utelåst på grund av tidigare fel inmatningar.
4. Repetera med fem och tio antal felaktiga lösenord.

WSTG-ATHN-04

Testscenario 1: Förbigå autentiseringssteg via URL

Förväntat utfall

Det går inte att komma åt en skyddad resurs om användaren inte är autentiserad.

Testdata

Användarnamn: *webadmin@example.com*
Lösenord: *flexauth2121*

Utförande

1. Logga in som *webadmin@example.com*
2. Klicka på Dashboard -> CMS - > Admin och spara URLen
3. Stäng ner fönstret och öppna ett inkognitofönster och klistra in den tidigare kopierade URLen.
4. Se om det nu går att komma åt de resurser som den inloggade användaren tidigare hade tillgång till.

WSTG-ATHN-05

Testscenario 1: Rimlig token-livslängd.

Förväntat utfall: Token byts ut eller tas bort efter en sedan tidigare konfigurerad tidsmängd.

Testdata

Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*
Access tokens livslängd sätts till 5 minuter i Okta.

Utförande

1. Logga in som *customer@example.com*
2. Öppna *Application*-fliken i Chrome DevTools.
3. Notera värdet på cookien “authenticated_user”.

4. Vänta i minst 5 minuter.
5. Uppdatera sidan i webbapplikationen.
6. Kontrollera att cookiens värde skiljer sig från värdet i steg 3.

Testscenario 2: Känslig data lagrad som klartext i cookie.

Förväntat utfall
Cookie innehåller inte känslig data i klartext.

Testdata
Användarnamn: *customer@example.com*
Lösenord: *flexauth2121*

Utförande
1. Logga in som *customer@example.com*
2. Öppna *Application*-fliken i Chrome DevTools.
3. Notera om cookien "authenticated_user" eller cookien ".sessionid" innehåller någon känslig data i klartext.

WSTG-ATHN-06

Testscenario 1: Bakåt-knappen återger inte sessionen

Förväntat utfall
Det går inte att komma att sessionen genom bakåt-knappen utan att behöva omautentisera sig.

Testdata
Användarkonto: *webadmin@example.com*
Lösenord: *flexauth2121*

Utförande:
1. Logga in på *https://localhost/episerver.*
2. Logga ut.
3. Försök nu komma åt sidan igen utan att autentisera genom att trycka på bakåtknappen i Chrome.

WSTG-ATHN-07

Testscenario 1: Registrering med vanliga lösenord

Förväntat utfall: Det går inte att skapa ett konto med ett vanligt lösenord

Testdata
Lösenord:
- 12345678
- 123456789
- picture1
- password
- password1
- qwerty

Utförande
1. Öppna en ny inkognitoflik och surfa till *https://localhost/account* . Klicka *Sign up.*
2. Fyll i valfri mail och försök skapa ett konto med samtliga vanliga lösenord ovan.

Testscenario 2: Registrering med samma lösenord som användarnamn

Förväntat utfall
Man kan inte använda sitt användarnamn som lösenord.

Testdata
N/A

Utförande

1. Navigera till *https://localhost/account*
2. Klicka *Sign up*
3. Fyll i valfri mail och försök skapa ett konto med samma lösenord som angiven mail.

WSTG-ATHN-08

Testscenario 1: Sätt säkerhetsfråga till användarens lösenord

Förväntat utfall: Det går inte att ha med sitt lösenord i ens säkerhetsfråga

Testdata
Användarnamn: *customer@example.com*
Lösenord: *flexauth2121*

Utförande

1. Logga in som *customer@example.com* i Okta.
2. Navigera till *end-user dashboard.*
3. Klicka på ditt **användarnamn** i sidhuvudet.
4. Klicka **Settings** och sedan **Edit** vid **Forgotten Password Question**.
5. Försök nu att mata in en säkerhetsfråga som innehåller ditt lösenord. T.ex. *"ditt lösenord är flexauth2121, svara 1234"*

Testscenario 2: Sätt säkerhetsfrågans svar till ett kort och enkelt svar

Förväntat utfall
Det går inte att att ha med ett svar med längd 1.

Testdata
N/A

Utförande

1. Logga in som *customer@example.com* i Okta.
2. Navigera till *end-user dashboard.*
3. Klicka på ditt **användarnamn** i sidhuvudet.
4. Klicka **Settings** och sedan **Edit** vid **Forgotten Password Question**.
5. Skriv in "what is 1 + 1" med svar 2.

Testscenario 3: Sätt säkerhetsfrågans svar till ett opersonligt svar.

Förväntat utfall
Det går inte att att ha med ett svar med längd 1.

Testdata
N/A

Utförande

1. Logga in som *customer@example.com* i Okta.
2. Navigera till *end-user dashboard.*
3. Klicka på ditt **användarnamn** i sidhuvudet.
4. Klicka **Settings** och sedan **Edit** vid **Forgotten Password Question**.
5. Skriv in “What is the capital of Sweden?” med svar “Stockholm”.

WSTG-ATHN-09

Testscenario 1: Ändra lösenord för en annan användare

Förväntat utfall

Det går inte att ändra en annan användares lösenord.

Utförande:

1. Öppna en inkognitoflik och surfa till *https://localhost/account*
2. Klicka på **Behöver du hjälp att logga in? -> Glömt lösenord?**
3. Skriv in *webadmin@example.com* som email och klicka återställ.
4. Kontrollera om något alternativ finns för att återställa lösenordet utan att ha tillgång till användarens mail.

WSTG-ATHN-10

Testscenario 1: Alternativa autentiseringskanaler

Förväntat utfall

Endast okta kan användas för autentisering.

Testdata

Användarnamn: *customer@example.com*
Lösenord: *flexauth2121*

Utförande

1. Försök komma åt följande sidor som kräver autentisering och kontrollera att bara Okta är en möjlig autentiseringskanal.
    - *https://localhost/episerver*
    - *https://localhost/account*
    - *https://localhost/util/login.aspx*

WSTG-ATHZ-02

Testscenario 1: Horizontal Authorization Bypass

Förväntat utfall

Användare A kan inte komma åt användare Bs resurs även om både A och B har samma privilegier och tillhör samma grupp genom att manipulera en request eller en cookie.

Testdata

Användarkonto A: *customer@example.com*
Användarkonto B: *customer2@example.com*
Lösenord: *flexauth2121*

Utförande:

1. Logga in med användarkonto A i en inkognitoflik i Google Chrome.
2. Logga in med användarkonto B i en inkognitoflik i Firefox.
3. Lägg till 12 antal av valfri produkt i varukorgen som användare A.
4. Försök ändra relevanta värden i autentiseringscookien tillhörande användarkonto A till värden från cookien tillhörande användarkonto B.
5. Notera om manipulation av cookie ger tillgång till den andra användarens resurser.

Testscenario 1: Vertical Authorization Bypass

Förväntat utfall

Användare A, *kunden*, kan inte komma åt EPiServer-panelen.

Testdata

Användarkonto A: *customer@example.com*
Användarkonto B: *administrator@example.com*
Lösenord: *flexauth2121*

Utförande:

1. Logga in i webbapplikationen som användare A i en inkognitoflik i Google Chrome.
2. Logga in i webbapplikationen som användare B i en inkognitoflik i Firefox.
3. Navigera till *https://localhost/episerver* som användare B.
4. Öppna *Inspector* i Firefox Development Tools som användare B och kopiera sessionscookiens värde.
5. Öppna *Application*-fliken i Chrome DevTools som användare A.
6. Klistra in värdet från steg 4 i sessionscookien användare A.
7. Uppdatera sidan som användare A.
8. Kontrollera om det går att komma åt *https://localhost/episerver* som användare A.

WSTG-ATHZ-03

Testscenario 1: URL Traversal

Förväntat utfall

Det finns inga sidor som tillåter en oautentiserad användare att komma åt känslig data.

Utförande

1. Öppna programmet Zap och starta spider med starting point på *https://localhost/*
2. Notera om det finns känslig data på några av sidorna sprider kunde få tillgång till.
3. Kör AJAX spider med starting point på *https://localhost/*
4. Notera om det finns känslig data på några av sidorna sprider kunde få tillgång till.
5. Starta spider med starting point på *https://localhost/episerver*
6. Notera om det finns känslig data på några av sidorna sprider kunde få tillgång till.
7. Kör AJAX spider med starting point på *https://localhost/episerver*

Testscenario 2: Manipulering av IP-adress

Förväntat utfall

Autentiseringstjänsten använder sig inte av detta fält för att hålla koll om misslyckade inloggningsförsök kommer från samma ip-adress.

Utförande

1. Navigera till inloggningskärmen för applikationen.
2. Öppna *Network*-fliken i Chrome DevTools och navigera till Network.
3. Rensa pakethistoriken i *Network*-fliken.

4. Försök logga in med felaktigt lösenord.
5. Kontrollera om huvudtypen "X-Forwarded-For" används i någon av förfrågningarna som gjorts hittills.
6. Om denna huvudtyp existerar, ändra fältet.
7. Undersök om upprepning av steg 4-6 kan kringgå autentiseringstjänstens utlåsningsmekanism.

Testscenario 3: Manipulering av grupptillhörighet via förfrågningar

Förväntat utfall
Ingen POST-metod har grupptillhörighet på ett modifierbart sätt.

Testdata
Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*

Utförande
1. Logga in i webbapplikationen med Zap Manual Explore.
2. Verifiera att ingen POST-metod innehåller ett fält där användaren kan skicka om metoden men ändrade värden för att hamna i en annan grupp med förhöjda rättigheter.

WSTG-ATHZ-04

Testscenario 1: Åtkomst av känslig data genom query-parametrar

Förväntat utfall
Ingen query-parameter på sidan ger tillgång till skyddade resurser som användaren i vanliga fall inte hade haft tillgång till.

Testdata
Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*

Utförande
1. Navigera till *https://localhost/account* och logga in som *customer@example.com*
2. Granska sidan efter användning av query-parametrar och om dessa kan ge tillgång till skyddade resurser.

WSTG-SESS-01

Testscenario 1: Slumpmässighet för autentiseringscookieslumpmässigheten

Förväntat utfall
Cookievärdena bedöms vara tillräckligt slumpmässiga av Burp Suites verktyg.

Testdata
Användarkonto 1: *defaultdefinederror@example.com*
Användarkonto 2: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Utförande
1. Hämta in minst 100 tokens från användarkonto 1 med hjälp av Burp Suite Sequencer.
2. Spara undan tokens för användarkonto 1 i en fil.
3. Upprepa steg 1 för användarkonto 2.
4. Spara undan tokens för användarkonto 2 i en annan fil.

5. Ladda in båda filerna med tokens i Burp Suite Sequencerns manuella läge.
6. Analysera tokens med Burps standardinställningar.
7. Kontrollera att slumpmässigheten bedöms vara *excellent*.

WSTG-SESS-02

Testscenario 1: Analys av attribut i autentiseringscookien

Förväntat utfall

Följande cookieattribut är definierade för autentiseringscookien.

- SameSite: Lax
- Secure: true
- HttpOnly: true
- Domain: localhost
- Path: /

Testdata

Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*

Utförande

1. Logga in i *https://localhost/account.*
2. Öppna *Application*-fliken i Chrome DevTools.
3. Kontrollera att attributen för cookien överensstämmer med det förväntade utfallet.

WSTG-SESS-03

Testscenario 1: Penetrationstest med Forced cookies

Förväntat utfall

Varken attacken vid steg 5 eller steg 11 kommer ha lagt till en produkt i offrets varukorg.

Testdata

Användarkonton:

- Offret: *defaultdefined@example.com*
- Attackeraren: *defaultdefinederror@example.com*
- Lösenord: *flexauth2121*

Utförande

1. Navigera till inloggningssidan.
2. Spara ett snapshot av alla cookies och deras värden.
3. Logga in som offret och navigera till en produktsida.
4. Sätt cookies och deras värden till snapshotten i steg 2.
5. Försök lägga till en produkt i varukorgen.
6. Rensa cookies och logga in som attackeraren.
7. Navigera till sidan i steg 3.
8. Sätt cookies och deras värden till snapshotten i steg 2.
9. Lägg till en produkten i varukorgen.
10. Rensa cookies och logga in som offret.
11. Kontrollera om produkten från steg 9 lagts till på offrets konto.

WSTG-SESS-04

Testscenario 1: Krypterad transport av sessionstokens

Förväntat utfall
Sessionscookie och autentiseringscookie skickas aldrig över en icke-krypterad kanal.

Testdata
Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Utförande
1. Öppna en webbläsarsession i ZAP och logga in med användarkontot.
2. Utför en spider-attack
3. Utför en AJAX spider-attack.
4. Kontrollera att alla förfrågningar eller responser med sessionstokens sker över HTTPS.

Testscenario 2: Cachning av känslig data
Förväntat utfall
Ingen caching görs på förfrågningar/svar med känslig information.

Testdata
Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Utförande
1. Utför steg 1-3 från testscenario 1.
2. Kontrollera att alla förfrågningar eller responser med känslig information inte cacheas (innehåller *Cache-Control: no-cache, Cache-Control: max-age=0* och *Expires: 0*)

WSTG-SESS-05
Testscenario 1: Dokumentstudie och kodgranskning av CSRF-skydd
Förväntat utfall
Okta har ett inbyggt skydd mot CSRF i sin autentiseringspipeline. Applikationen implementerar grundläggande skyddsmekanismer mot CSRF.

Testdata
N/A

Utförande
1. Undersöka Oktas dokumentation för referenser till “CSRF” eller “Cross Site Request Forgery”.
2. Granska koden i applikationen, däribland användning av SameSite-attribut för cookies.

Testscenario 2: Penetrationstest med CSRF - Tillägg i varukorg
Förväntat utfall
Tillägget registeras inte i den inloggade användarens varukorg.

Testdata
Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Kod enligt blocket nedan (*se block WSTG-SESS-05-1*)

```
<html>

<body>
    <form action="https://localhost/cart/Update?code=T-Shirt-L_1&quantity=1"
method="POST">
        <input type="hidden" name="email" value="pwned@evil-user.net" />
    </form>
    <script>
        document.forms[0].submit();
    </script>
</body>

</html>
```

Block WSTG.SESS-05-1: *Simpelt CSRF-attackformulär.*

Utförande

1. Logga in i applikationen.
2. Lägg till en produkt i varukorgen.
3. Notera varukorgens innehåll.
4. Öppna upp attackformen från testdata i samma webbläsare.
5. Kontrollera att ytterligare produkt inte lagts till i den inloggade användarens varukorg.

WSTG-SESS-06

Då applikationen inte är färdigdesignad testas inte gränssnittet enligt **Testing for Log Out User Interface.** Detta är något som bör kontinuerligt tänkas på under applikationens fortsatta utveckling.

Sessionsterminering testas i WSTG-SESS-07.

Testscenario 1: Sessionsterminering vid utloggning

Förväntat utfall

Cookien som användes under sessionen är inte längre giltig efter användaren har loggat ut.

Testdata

Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Utförande

1. Logga in med användarkontot.
2. Spara undan autentiseringscookiens värde.
3. Logga ut användaren.
4. Verifiera att sidor som kräver autentisering inte kan kommas åt.
5. Modifiera autentiseringscookiens värde till värdet från steg 2.
6. Verifiera att användaren inte autentiseras.

WSTG-SESS-07

Testscenario 1: Sessionsterminering vid inaktivitet

Förväntat utfall

Omautentisering krävs när man efter en, i förväg inställd, mängd minuter navigerar till en sida som kräver autentisering.

Testdata

Livstiden för en refresh token sätts till 10 minuter från senaste användning, i Okta.

Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Utförande

1. Logga in med användarkontot.
2. Låt webbläsaren stå orörd i minst 10 minuter.
3. Navigera till en sida som kräver autentisering.
4. Verifiera att omautentisering krävs.

WSTG-SESS-08

Testscenario 1: Registrera användare session puzzle

Förväntat utfall En sessionsvariabel för en viss användare sätts inte när man försöker skapa en användare med dennes redan existerande email.

Testdata
Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*

Utförande

1. Öppna en ny inkognitoflik i Google Chrome och klicka på **registrera dig.**
2. Öppna en ny inkognitoflik i samma fönster och surfa till *https://dev-6673066.okta.com/enduser/settings* och logga in som *customer@example.com*
3. Gå tillbaka till registreringsfliken och skriv in ett känt admin-användarnamn(t.ex. *webadmin@example.com*) och fyll i valfritt namn och lösenord. Klicka **Registrera.**
4. Gå tillbaka till den inloggade fliken och uppdatera sidan för att kontrollera att du inte fått tag på webadmins session under registreringsförsöket.
5. Testa komma åt sidan *https://localhost/episerver*

Testscenario 2: Ändra lösenord session puzzle

Förväntat utfall
En sessionsvariabel för en viss användare sätts inte när en attackerare försöker använda Oktas lösenordsändringsfunktion.

Testdata
Användarnamn: *customer@example.com*
Lösenord: *flexauth2121*

Utförande

1. Öppna en ny inkognitoflik i Google Chrome och klicka på **Glömt lösenord.**
2. Öppna en ny inkognitoflik i samma fönster och surfa till *https://dev-6673066.okta.com/enduser/settings* och logga in som *customer@example.com*
3. Gå tillbaka till glömt lösenord-fliken och skriv in ett känt admin-användarnamn(t.ex. *webadmin@example.com*). Klicka **Återställ via epost.**
4. Gå tillbaka till den inloggade fliken och uppdatera sidan för att kontrollera att du inte fått tag på webadmins session under återställningsförsöket.
5. Kontrollera om det går att komma åt sidan *https://localhost/episerver*

Testscenario 3: Sök upp formulär där en användare kan fylla i användarinformation.

Förväntat utfall

Det finns inga formulär där användarinformation kan fyllas i

Testdata
Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*
Utförande
1. Öppna en ny inkognitoflik och surfa till *https://localhost/account*. Logga in som *customer@example.com*.
2. Navigera runt bland samtliga sidor efter forms som man kan mata in t.ex email, användarnamn eller lösenord.

WSTG-SESS-09

Testscenario 1: Omdirigering från HTTP till HTTPS

Förväntat utfall
Omdirigering till HTTPS görs för alla förfrågningar.

Testdata
Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Utförande
1. Starta en ZAP-webbläsarsession.
2. Logga in i applikationen med användarkontot.
3. Utför en Spiderattack.
4. För varje förfrågan i historiken i ZAP, försök komma åt URLen med HTTP-protokollet.
5. Verifiera att alla förfrågningar omdirigeras till HTTPS.

WSTG-CRYP-01

Testscenario 1: Säkra Certifikat

Förväntat utfall
Certifikatet är säkert.

Testdata
N/A

Utförande
Kontrollera servercertifikatet efter följande:
1. Nycklar som används till certifikatet är minst 2048 bitar långa.
2. Servercertifikatet är minst SHA-256. MD5 eller SHA-1 räknas t.ex som osäkra här.
3. Att detta är inom sin valideringsperiod.
4. Om detta är skapat efter 1 september 2020 ska det inte ha en längre lifespan än 393 dagar.
5. Att detta är signerat av en trusted certificate authority.
6. Att detta har en SAN som matchar hostname för systemet.

Testscenario 2: Kontrollera kryptering i Nmap

Förväntat utfall
Ingen osäker cipher suite används och det använda certifikatet upprätthåller en bra säkerhetsstandard.

Testdata

N/A

Utförande

1. Starta programmet Nmap.
2. Kör scriptet “$ -p 443 --script ssl-cert nmap localhost”
3. Kontrollera att certifikatet är enligt specifikation i scenario 1.
4. Kör scriptet “$ -p 443 --script +ssl-enum-ciphers nmap localhost”
5. Kontrollera att ingen osäker cipher suite används. Exempel på osäkra är DES eller SHA-1. Betyg bör vara B eller Högre.
6. Kontrollera att ingen förlegad TLS version(1.0 och 1.1) används.

Testscenario 3: Påtvinga HTTP-protokoll

Förväntat utfall

Det går inte att komma åt sidan via HTTP

Testdata

N/A

Utförande

1. Öppna google chrome och försök komma åt sidan via http://localhost
2. Om du blir omdirigerat till https när du försöker komma åt sidan via http måste man försäkra sig om att sidan är tillagd i “preload list”. Mer info om sidan uppfyller kraven för detta på *https://hstspreload.org/*.

WSTG-CRYP-02

Testscenario 1: Padding oracle

Förväntat utfall

Det finns säkerhetsåtgärder i web.config för att se till att hackare inte kan använda sig av ASP.NETs felmeddelande för att bryta krypteringens padding.

Utförande

1. Kontrollera att fältet errorMode är satt till “Custom” i Sites web.config
2. Kontrollera att följande rad finns med i Sites web.config under *<system.web>:*
   <customErrors mode="On" redirectMode="ResponseRewrite" defaultRedirect="~/error.aspx" />

WSTG-CRYP-03

Testscenario 1: Lösenord eller krypteringsnycklar i klasser eller konfig-filer.

Förväntat utfall

Det finns inga lösenord eller krypteringsnycklar i klartext i någon projektfil.

Testdata

N/A

Utförande

1. Öppna visual studio och sök i filer (ctrl + shift + f) efter följande termer i hela projektmappen: password | pwd |user | guest| admin | encry | key | decrypt | sharekey.
2. Kontrollera om det finns några lösenord eller krypteringsnycklas någonstans i projektet.

Testscenario 2: Secure session cookies

Förväntat utfall
Samtliga sessionscookies är av typen Secure.

Testdata
Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*

Utförande

1. Logga in på *https://localhost/account*
2. Öppna Application-fliken i Chrome DevTools och kontrollera att alla cookies är säkra enligt deras "Secure"-fält.

WSTG-CRYP-04

Testscenario 1: Säkra krypteringsmetoder i källkoden

Förväntat utfall: Inga osäkra krypteringsmetoder används.

Testdata
N/A

Utförande

1. Öppna visual studio och sök efter följande termer: (MD4, MD5, RC4, RC2, DES, Blowfish, SHA-1, ECB). Om dessa används till någon väsentlig kryptering eller signering bör dessa bytas ut till säkrare varianter.

WSTG-CLNT-01

Testscenario 1: Källkodsgranskning för DOM-based XSS

Enligt http://www.webappsec.org/projects/articles/071105.html

Förväntat utfall
Det finns inga potentiellt farliga HTML-metoder i clientkoden.

Testdata
N/A

Utförande

1. Surfa till *https/localhost/* och öppna chrome dev tools
2. Klicka på **Sources,** högerklicka på **localhost** och välj **Search in all files.**
3. Sök efter följande
    - document.URL
    - document.URLUnencoded
    - document.location
    - document.referrer
    - window.location
4. Notera om dessa används eller inte. I fallen där dessa används ska testaren kontrollera om värdet som returneras från dessa metoder används i DOM eller HTML-ändrande metoder som t.ex. document.write(…), document.writeln(…) eller document.body.innerHtml

WSTG-CLNT-02

Hänvisas till testscenarion under WSTG-CLNT-01 och WSTG-CLNT-03 som täcker säkerhetstester för detta id. Webbapplikationen är i detta skede inte tillräckligt utvecklad för att göra någon större granskning av sidor där farlig javascript-kod kan injiceras. Det kan vara av intresse för företaget att återbesöka detta och designa detta test närmare lansering.

Godkänt/Underkänt
Godkänt.

WSTG-CLNT-03

Testscenario 1: JavaScript-funktioner som är sårbara för HTML-injektioner

Förväntat utfall
Inga utsatta JavaScript-funktioner används tillsammans med ovaliderad användardata.

Testdata
N/A

Utförande

1. Kodgranska frontend efter referenser till
    a. *location*
    b. *document.write()*
    c. *document.writeln()*
    d. *outerHTML*
    e. *innerHTML*

    som används tillsammans med data försedd av användaren.

2. Om användardata påträffas tillsammans med ovan nämnda referenser, kontrollera att användardata går genom lämplig encoding eller validering.

Testscenario 2: HTML-injektion i URL-queryparametrar.

Förväntat utfall
Sidan godtar inte den injicerade HTML-koden.

Testdata
HTML-kod 1:
<script>console.log('injection');</script>
HTML-kod 2:
%3Cscript%3Econsole.log(%E2%80%98injection%E2%80%99)%3B%3C%2Fscript%3E

Utförande

1. Navigera till applikationen
2. Sök på HTML-kod 1 och 2 i sökfältet på sidan.
3. Manipulera URL:en direkt med kod 1 och 2.
4. Notera resultatet.

WSTG-CLNT-04

Testscenario 1: Omdirigering med window.location-objektet.

Förväntat utfall
Inga omdirigeringar görs med window.location-objektet.

Testdata

N/A

Utförande

1. Gör en textsökning efter “window.location” i alla filer som innehåller frontend-kod.
2. Kontrollera att substrängar av objektet inte används för att omdirigera användaren.

WSTG-CLNT-05

Testscenario 1: CSS injection med Nmap.

Förväntat utfall: Webbapplikationen är inte sårbar för CSS injections.

Testdata

N/A

Utförande

1. Starta programmet nmap.
2. Kör skriptet “nmap -p 443 --script ssl-ccs-injection localhost”
3. Notera eventuella meddelanden om applikationens sårbarhet.

WSTG-CLNT-06

Testscenario 1: Identifiering av attackytor för resursmanipulation i frontend

Förväntat utfall

Inga attackytor med icke-validerad inmatad data identifieras.

Testdata

N/A

Utförande

1. Navigera till *https://localhost*
2. Öppna i Chrome DevTools
3. Sök efter följande taggar/objekt och kontrollera att ingen lämnar möjlighet för att injicera en annan sidas skadliga kod, t.ex. scripts, utan att dessa valideras:
    - Iframe
    - A
    - xhr.open(method, \[url\], true);
    - Link
    - Img
    - Object
    - script

WSTG-CLNT-07

Testscenario 1: Säker användning av CORS

Förväntat utfall

Det finns ingen känslig data i svaren på de requests som har Access-Control-Allow-Origin fältet satt till *.

Testdata

Användarkonto: *customer@example.com*
Lösenord: flexauth2121

Utförande

1. Öppna programmet Zap
2. Klicka på **Quick start -> manual explore**
3. Skriv in *https://localhost/account* i URL to explore.
4. Klicka på **Launch browser.**
5. Logga in som *customer@example.com*
6. Klicka på din cart längst uppe till höger.
7. Gå tillbaka till Zap och klicka på **Alerts-**fliken
8. Undersök om det finns några varningar angående *Cross-Domain Misconfiguration*
9. Om så är fallet, undersök varje HTTP-requests response för att kontrollera om det finns känslig data is svaret.

WSTG-CLNT-08

Testscenario 1: Användning av sårbara ActionScript-funktioner

Förväntat utfall

Ingen användning kan hittas och applikationen bedöms vara skyddad mot Cross Site Flashing.

Testdata

N/A

Utförande

1. Granska samtliga JavaScript, TypeScript och C#-källkodsfiler i jakt på användning av Flash.
2. Notera alla referenser till funktionerna nedan:
    a. loadVariables()
    b. loadMovie()
    c. getURL()
    d. loadMovieNum()
    e. FScrollPane.loadScrollContent()
    f. LoadVars.load
    g. LoadVars.send
    h. XML.load('url')
    i. LoadVars.load('url')
    j. Sound.loadSound('url' , isStreaming);
    k. NetStream.play('url');
    l. flash.external.ExternalInterface.call(_root.callback)
    m. htmlText
3. Kontrollera att inga referenser hittas i steg 2.

WSTG-CLNT-09

Testscenario 1: Penetrationstest med clickjacking

Förväntat utfall

Attacken misslyckas eftersom X-Frame-Options-svarshuvudet är satt till SAMEORIGIN och förhindrar nästling av sidan i en iframe från sidor som inte tillhör samma domän.

Testdata

HTML-sida med kod enligt block nedan (*se block WSTG-CLNT-09-1)*.

```
<html>

<head>
```

```
    <title>Clickjack test page</title>
</head>

<body>
    <iframe src="https://localhost" width="500" height="500"></iframe>
    <iframe src="https://localhost/account" width="500" height="500"></iframe>
    <iframe src="https://localhost/cart" width="500" height="500"></iframe>
    <iframe src="https://localhost/episerver" width="500" height="500"></iframe>
</body>

</html>
```

Block WSTG-CLNT-09-1. *Simpel testsida för clickjack-attacker.*

Utförande

1. Öppna HTML-sidan innehållandes koden från testdatan.
2. Verifiera att alla iframes får sin anslutning avvisad.

WSTG-CLNT-10

Testscenario 1: Användning av WebSockets.

Förväntat utfall

WebSockets används inte i applikationen.

Testdata

N/A

Utförande

1. Starta en manuell skanningsession på *https://localhost* i OWASP ZAP.
2. Logga in i systemet.
3. Utför en Spider-attack och en Ajax Spider-attack.
4. Granska WebSocket-historiken för att påvisa användning av WebSockets.

WSTG-CLNT-11

Testscenario 1: Användning av Cross Document Messaging

Förväntat utfall

Om Cross Document Messaging används, uppfyller användning de specificerade säkerhetskriterierna nedan.

Testdata

Kriterier för användning av postMessage

- Mottagande domäner är explicit specifierade.
- ‘*’ används aldrig som andra-parameter i postMessage()
- addEventListeners som mottar meddelanden från andra domäner verifieras alltid innan data börjar manipuleras.

Kriterier för användning av eventListener(“message”, …):

- Ingående data använder inte eval()-metod för evaluering.
- Ingående data tillsätts inte med innerHTML-propertyn.
- Ingående datas domäner valideras explicit, inte enbart som delsträngar.

Utförande

1. Undersök koden för referenser till postMessage()-metoden.
2. Om referenser hittas i steg 1, validera efter kriterier för metoden enligt relevant testdata.

3. Undersök koden för referenser till addEventListener(‘message’, …).
4. Om referenser hittas i steg 3, validera efter kriterier för metoden enligt relevant testdata.

WSTG-CLNT-12

Testscenario 1: Känslig data lagrad i webbläsare

Förväntat utfall
För de lagringsmekanismer som används lagras ingen känslig data i webbläsaren. Web SQL används inte.

Testdata
Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

Utförande
1. Logga in på *https://localhost/account* med användarkontot.
2. Lägg till en produkt i varukorgen.
3. Navigera till varukorgssidan */cart*.
4. Öppna *Application* > *Storage* i Chrome DevTools.
5. Verifiera att följande lagringsmekanismer inte läcker användarkontoinformation.
    a. Local Storage
    b. Session Storage
    c. IndexedDB
    d. Cookies
6. Verifiera att Web SQL inte används.

Testscenario 2: Känslig data lagrad i window-objektet

Förväntat utfall
Ingen användarinformation lagras globalt i Window-objektet.

Testdata
Användarkonto: *defaultdefined@example.com*
Lösenord: *flexauth2121*

JavaScript-kod enligt block nedan (*se block WSTG-CLNT-12-1)*

```
(() => {
  const iframe = document.createElement('iframe');
  iframe.style.display = 'none';
  document.body.appendChild(iframe);
  const currentWindow = Object.getOwnPropertyNames(window);
  const results = currentWindow.filter(
    prop => !iframe.contentWindow.hasOwnProperty(prop)
  );
  document.body.removeChild(iframe);
  results.forEach(key => console.log(`${key}: ${window[key]}`));
})();
```

Block WSTG-CLNT-12-1. *Simpel JavaScript-snippet för analys av window-objektet.*

Utförande
1. Utför steg 1-4 från testscenario 1.
2. Kopiera och kör kodsnippeten från testdata.
3. Verifiera i konsolfönstret att ingen användardata skrivs ut.

WSTG-CLNT-13

Testscenario 1: Dataläckage i JavaScript- eller JSON-responser.

Förväntat utfall
Ingen känslig användardata läcks genom responserna. DetectDynamicJS rapporterar inga fel.

Testdata
Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*

Utförande

1. Verifiera att pluginskriptet DetectDynamicJS är laddat i Burp Suite.
2. Starta en webbläsare i Burp Suite.
3. Navigera till och logga in i webbapplikationen med användarkontot.
4. Navigera till *https://localhost/episerver*
5. Navigera till *https://localhost/*
6. Sök upp och lägg till en produkt i varukorgen.
7. Navigera till varukorgen.
8. Öka och minska produktantalet i varukorgen.
9. Ta bort produkten ur varukorgen.
10. Logga ut användaren.
11. Upprepa steg 5-9.
12. Granska responser i *Burp Suite > Logger* efter läckt användardata.
13. Verifiera att DetectDynamicJS inte rapporterat några fel.

Testscenario 2: Dataläckage via globala funktioner.

Förväntat utfall
Inga funktioner som kan returnera användardata upptäcks.

Testdata
Se scenario 1.

Utförande

1. Utför WSTG-CLNT-12, testscenario 1 med tillhörande testdata, steg 1-2.
2. Granska globala funktioner som returneras och deras förmåga att eventuellt komma åt känslig användardata.

Testscenario 3: Dataläckage via CSV with Quotations Theft.

Förväntat utfall
Om CSV-responser rapporteras, riskerar de inte läcka känslig data vid en injektionsattack.

Testdata
Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*

Utförande

1. Utför en spiderattack på webbapplikationen med ZAP, alternativt genomsök hela webbapplikationen med en Burp Suite Proxy.
2. Upprepa steg 1 som inloggad användare. Använd användarkontot i testdata.

3. Granska HTTP-historiken efter responser vars Content-Type är **text/csv** eller **application/csv.**
4. Om sådana responser hittas, verifiera att injektionssänkor inte riskerar läcka känslig data.

Testscenario 4: Dataläckage via Prototype Chaining med nyckelordet this

Förväntat utfall
Inga objekt som kan nås via prototype chaining läcker känslig data.

Testdata
Användarkonto: *customer@example.com*
Lösenord: *flexauth2121*

HTML-kod enligt block nedan (*se block WSTG-CLNT-13-1).*

```
 <div id="result">

</div>
<script>
    var resultString = "Prototype chain values found: <br>";
    Array.prototype.forEach = function (callback) {
        resultString += "<b>"
        for (var i = 0, length = this.length; i < length; i++) {
            if (i > 0) {
                resultString += ", ";
            }
            resultString += this[i];
        }
        resultString += "</b>";
        var div = document.getElementById("result");
        div.innerHTML = resultString;
        if(this.length > 0) resultString += "<br>"
    };
</script>

<script src="JS-FILE-URL"></script>
```

Block WSTG-CLNT-13-1. *HTML-sida med överskriven Array.prototype.forEach-funktion.*

Utförande
1. Utför en spiderattack på webbapplikationen med ZAP i en inloggad session, alternativt genomsök webbapplikationen med en Burp Suite Proxy.
2. Identifiera alla responser som returnerar en JavaScript-fil från webbapplikationens domän.
3. Utgå från koden i *block WSTG-CLNT-13-1*. För varje fil som identifieras i steg 2, lägg till en scripttagg där scriptkällan *JS-FILE-URL* ersätts med js-filens url..
4. Öppna en HTML-fil innehållandes koden som skapades i steg 3.
5. Granska resultatet efter känslig data.

Bilaga 5. Testresultat, utfärdad 2021-05-05

WSTG-IDNT-01

🟩 Testscenario 1: Dokumentation av roller som används i applikationen

Faktiskt utfall
Rollnamn är definierade.
I autentiseringsmodulen och Okta finns rollerna

- Customers
- Everyone (standardgrupp som alltid finns).

I applikationen finns rollerna

- CommerceAdmins
- Administrators
- WebAdmins
- WebEditors
- Everyone

Godkänt/Underkänt
GODKÄNT

🟩 Testscenario 2: Ändra användarroller genom manipulation av sessionstoken

Faktiskt utfall
Cookien är krypterad och inga särskilda värden kan urskiljas. Man kan inte enkelt ändra cookien för att få tillgång till andra roller.

Godkänt/Underkänt
GODKÄNT

🟨 Testscenario 3: Se över granulariteten i rollernas rättigheter.

Faktiskt utfall
Då miljön ännu inte har några konkret slutgiltiga roller är det omöjligt att komplett utreda granulariteten hos deras rättigheter. Rollerna i autentiseringstjänsten används i nuläget inte, utan tjänsten autentiserar endast användare. Sedan sköter rollfördelning av EPiServer vars standardroller är breda.

Godkänt/Underkänt
GODKÄNT med anmärkning

Det finns framtida möjlighet att mappa roller från Okta till EPiServer, eller att låta Okta helt sköta rollfördelningen. Då är det av största vikt att Principle of Least Privilege följs, och användarrollerna ses över och definieras utefter den framtida kundens behov.

WSTG-IDNT-02

🟩 Testscenario 1: Registrering enligt säkerhetskrav
Faktiskt utfall
Registrering av konto asdasdasd@example.com och lösenord enligt testdata lyckas.

Godkänt/Underkänt
GODKÄNT

🟩 Testscenario 2: Registrering som bryter säkerhetskrav
Faktiskt utfall
Registrering av konto peter@example.com och lösenord *pic1* misslyckas.
Registrering av konto peter@example.com och lösenord *peter123* misslyckas.
Registrering av konto peter@example.com och lösenord *password123* misslyckas.
Användaren peter@example.com är inte tillagd i Oktas användarkatalog.

Godkänt/Underkänt
GODKÄNT

🟩 Testscenario 3: Registrering av redan registrerad e-postadress
Faktiskt utfall
Registrering av konto defaultdefined@example.com misslyckas.

Godkänt/Underkänt
GODKÄNT

WSTG-IDNT-03

🟩 Testscenario 1: Åtkomst till administratörsfunktion i Okta som kund
Faktiskt utfall
Användaren customer@example.com kommer inte åt Oktas administratörsfunktioner via manuell navigering eller inklistring av URI.

Godkänt/Underkänt
GODKÄNT

🟩 Testscenario 2: Åtkomst till administratörsfunktion i Okta som administratör
Faktiskt utfall
Som förväntat för autentiseringstjänsten. Okta har en hög upplösning på administratörsroller vars rättigheter tydligt sammanfattas i gränssnittet. Endast administratörer i Okta-domänen kan skapa nya roller.

Godkänt/Underkänt
GODKÄNT

WSTG-IDNT-04

🟩 Testscenario 1: Uppräkning av registrerade användare via inloggningsformulär
Faktiskt utfall

Ingen skillnad mellan responserna av inloggningsförsöken med *defaultdefined@example.com* och *notreg@example.com* kan urskiljas. Attackeraren kan inte bekräfta om ett konto finns registrerat via inloggningsformuläret.

Godkänt/Underkänt
GODKÄNT

🟥 Testscenario 2: Uppräkning av registrerade användare via registreringsformulär
Faktiskt utfall
Inloggning med *defaultdefined@example.com* ger ett felmeddelande som avslöjar att emailadressen är registrerad.

Godkänt/Underkänt
UNDERKÄNT

WSTG-IDNT-05

🟥 Testscenario 1: Granskning av stöd för säkerhetskrav på användarnamn
Faktiskt utfall
Någon sådan funktion existerar inte, användarnamn måste vara användarens e-postadress.

Godkänt/Underkänt
UNDERKÄNT

WSTG-ATHN-01

🟩 Testscenario 1: Känslig data okrypterat.
Faktiskt utfall
Analys av trafik med Wireshark ger till störst del en mängd UDP-paket som alla innehåller krypterad data där ingen känslig information kan utläsas. Vidare sökning för termer som "username", "email", "password", "okta", "token" ger inga fynd.

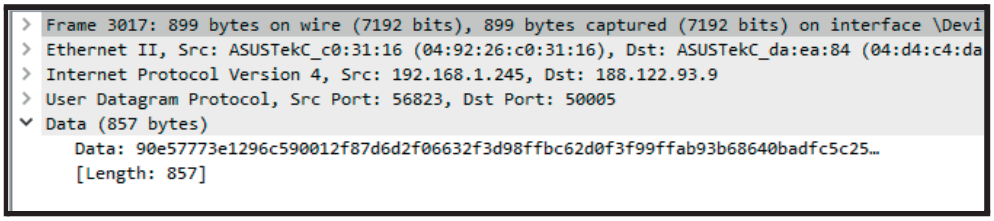


Bild WSTG-ATHN-01-1. *Exempel på UDP-paket vid inloggning i webbapplikationen.*

Godkänt/Underkänt
GODKÄNT

WSTG-ATHN-02

🟩 Testscenario 1: Defaultvärden på nyskapade konton
Faktiskt utfall

Inga defaultvärden finns för inloggningsdetaljer. Vid registrering måste en användare fylla i giltigt lösenord och giltig e-postadress.

Godkänt/Underkänt
GODKÄNT

WSTG-ATHN-03

🟩 Testscenario 1: Utlåsning vid upprepade felaktiga lösenordsförsök
Faktiskt utfall
Kontot låser sig efter tio försök med felaktigt lösenord. Detta är i enlighet med standardinställningarna i Okta.

Godkänt/underkänt
GODKÄNT.

WSTG-ATHN-04

🟩 Testscenario 1: Förbigå autentiseringssteg via URL
Faktiskt utfall
Användaren får inte åtkomst till skyddade resurser enbart genom inklistring av URI från en berättigad session.

Godkänt/underkänt
GODKÄNT.

WSTG-ATHN-05

🟩 Testscenario 1: Rimlig token-livslängd.
Faktiskt utfall
Efter fem minuters inaktivitet, uppdateras cookiens värde i enlighet med access tokens livslängd.

Godkänt/underkänt
GODKÄNT

🟩 Testscenario 2: Känslig data lagrad som klartext i cookie.
Faktiskt utfall
Sessionscookien eller autentiseringscookien innehåller inte någon känslig data i klartext. Den innehåller inte heller information om vilken kryptering eller encoding som använts för att skapa värdet.

Godkänt/underkänt
GODKÄNT

WSTG-ATHN-06

🟩 Testscenario 1: Bakåt-knappen återger inte sessionen
Faktiskt utfall

Okta tvingar användaren att omautentisera om denna har loggat ut. Det går inte att komma åt sessionen igen genom att trycka på bakåtknappen.

Godkänt/underkänt
GODKÄNT

WSTG-ATHN-07

🟩 Testscenario 1: Registrering med vanliga lösenord
Faktiskt utfall
Det går inte att skapa ett konto med något av lösenorden som listas under testdata.

Godkänt/underkänt
GODKÄNT

🟩 Testscenario 2: Registrering med samma lösenord som användarnamn
Faktiskt utfall
En del av användarnamnet får inte finnas i lösenordet.

Godkänt/underkänt
GODKÄNT

WSTG-ATHN-08

🟥 Testscenario 1: Sätt säkerhetsfråga till användarens lösenord
Faktiskt utfall
Säkerhetsfrågan kan innehålla lösenordet om frågan inkluderar minst ett annat tecken.
Exempelvis om lösenordet är *flexauth2121*:

- *"flexauth2121"* godkänns inte.
- *"_flexauth2121_"* godkänns.

Godkänt/underkänt
UNDERKÄNT

🟩 Testscenario 2: Sätt säkerhetsfråga till ett kort och enkelt svar.
Faktiskt utfall
Det går inte att ha en säkerhetsfråga vars svar är mindre än 4 tecken.

Godkänt/underkänt
GODKÄNT

🟥 Testscenario 3: Sätt säkerhetsfrågans svar till ett opersonligt svar.
Faktiskt utfall
Då användarskapade frågor är tillåtna, kan frågor med opersonliga svar skapas.

Godkänt/underkänt
UNDERKÄNT

WSTG-ATHN-09

🟩 Testscenario 1: Ändra lösenord för en annan användare

Faktiskt utfall:
Utan tillgång till användarens mailkonto är det omöjligt att ändra en annan persons mail via Oktas inloggningsskärm.

Godkänt/underkänt:
GODKÄNT

WSTG-ATHN-10

🟩 Testscenario 1: Undersök alternativa autentiseringskanaler

Faktiskt utfall
/episerver och */account* sidor kräver autentisering med Okta.
/util/login.aspx autentiserar inte längre användare, då EPiServers standardautentisering inte längre används.

Godkänt/Underkänt
GODKÄNT

WSTG-ATHZ-02

🟩 Testscenario 1: Horizontal Authorization Bypass

Faktiskt utfall
Endast en total kopiering av autentiseringscookien kommer ge användare A tillgång till resurser som tillhör användare B. Detta är inte görbart såvida inte personen har fri tillgång till båda webbläsarsessionerna, på samma maskin.

Enskilda värden i cookien kan inte urskiljas eller manipuleras meningsfullt. Således är det inte ett hot.

Godkänt/Underkänt
GODKÄNT

🟩 Testscenario 1: Vertical Authorization Bypass

Faktiskt utfall
Användare A får inte tillgång till */episerver* som användare B har tillgång till, enbart genom att kopiera sessionsvärdet för användare B. Det vill säga, en användares rättigheter har inte att göra med dennes sessionsvariabler i backend.

Godkänt/Underkänt
GODKÄNT

WSTG-ATHZ-03

🟩 Testscenario 1: URL Traversal

Faktiskt utfall
Ingen känslig data fås genom användning av spider mot någon av adresserna.

Godkänt/Underkänt
GODKÄNT

🟩 Testscenario 2: Manipulering av IP-adress
Faktiskt utfall
Inga huvuden av typen *X-Forwarded-For* finns under inloggningsförsök. Således kan inte utlåsningsmekanismen kringgås.

Godkänt/Underkänt
GODKÄNT

🟩 Testscenario 3: Manipulering av grupptillhörighet via förfrågningar
Faktiskt utfall
Inga grupp- eller rolltillhörighetsfält identifieras i POST-förfrågningar under inloggning.

Godkänt/Underkänt
GODKÄNT

WSTG-ATHZ-04

🟩 Testscenario 1: Åtkomst av känslig data genom query-parametrar
Faktiskt utfall
Den enda funna queryparametern i webbapplikationen är i nuläget de som relaterar till sökformuläret. Denna riskerar inte röja känslig data.

Godkänt/Underkänt
GODKÄNT

WSTG-SESS-01

🟨 Testscenario 1: Slumpmässighet för autentiseringscookieslumpmässigheten
Faktiskt utfall
Som förväntat. Stickprovet är för litet för att vara helt tillförlitligt, men på grund av API-begränsningar i gratisplanen för Okta testas inte slumpmässigheten för fler cookies. Slumpmässigheten bedöms vara *excellent.* Skärmdump av analysens resultat finns nedan (*se bild WSTG-SESS-01-1 nedan*).

Bild WSTG-SESS-01-1. Skärmdump av analysresultat.

Godkänt/Underkänt
GODKÄNT med anmärkning:

Testa med ett större urval när tillgång till fler API-anrop inte riskerar låsa Okta-domänen.

WSTG-SESS-02

🟨 Testscenario 1: Analys av attribut i autentiseringscookien
Faktiskt utfall
Autentiseringscookien har attribut enligt bild *WSTG-SESS-02-1* nedan.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
authenticated_user	Bt2NP...	localhost	/	Session	1896	✓	✓	Lax

Bild WSTG-SESS-02-1. *Autentiseringscookie med attribut, inspekterad i Chrome DevTools*

Godkänt/Underkänt
GODKÄNT med anmärkning

*Vid lansering **ska** Domain-attribut konfigureras till önskad domän.*

WSTG-SESS-03

🟩 Testscenario 1: Penetrationstest med Forced cookies

Faktiskt resultat

Ingen av attackerna lyckas och offrets kundkorg är bevarad. Information om den inloggade användaren fixeras inte till sessionen i sig, utan till den autentiseringscookie man har. Således skiljer de sig garanterat mellan inloggat och icke-inloggat läge.

Godkänt/Underkänt
GODKÄNT

WSTG-SESS-04

🟩 Testscenario 1: Krypterad transport av sessionstokens

Faktiskt resultat

Spideringresultat visar att alla förfrågningar och responser med sessionstokens över HTTPS. Loggfil uteblir då namngivning på vissa resurser faller under sekretess.

Godkänt/Underkänt
GODKÄNT

🟥 Testscenario 2: Cachning av känslig data

Faktiskt resultat

Spideringresultat ger responser där svarshuvudtyperna inte har önskade attribut. Dock rör det sig inte om cachning av känslig data i nuläget. Sökformuläret cacheas till exempel enligt *Cache-Control: private*, men detta kan anses vara relativt ofarligt.

Godkänt/Underkänt
UNDERKÄNT med anmärkning

Då funktionalitet är begränsad i nuläget är det av största vikt att detta test utförs så fort applikationen får tillgång till mer användarinmatad data, såsom kreditkortsuppgifter eller dylikt. Sådan känslig information måste granskas för sårbarheter i cachelagring.

Underkänt betyg i nuläget avser inte att systemet läcker särskilt känslig användardata i nuläget. Det avser snarare understryka vikten av fortsatt testning.

WSTG-SESS-05

🟨 Testscenario 1: Dokumentstudie och kodgranskning av CSRF-skydd

Faktiskt utfall

Som förväntat. Oktas autentisering kräver state-parameter för interaktion med /authorize-endpointen i deras API, vilket förhindrar CSRF-attacker under inloggning.

I applikationen valideras autentiseringscookies med SameSite=Lax, vilket helt skyddar mot POST-baserade CSRF-attacker.

Godkänt/Underkänt

GODKÄNT med anmärkning

I nuläget använder inte applikationen GET-förfrågningar som förändrar tillstånd för användare, således är SameSite=Lax tillräckligt.

🟨 Testscenario 2: Penetrationstest med CSRF - Tillägg i varukorg

Faktiskt utfall

Som förväntat. Cookies använder *SameSite=Lax,* vilket hindrar browsern från att inkludera cookien när den kommer från ett annat origin vid en POST-request.

Godkänt/Underkänt

GODKÄNT med anmärkning

Detta skydd finns ej vid GET-förfrågningar. Applikation bör inte orsaka tillståndsförändringar vid GET-förfrågningar. Om så är fallet, måste en Strict SameSite-policy användas, vilket i många fall försämrar och försvårar en bra användarupplevelse.

WSTG-SESS-06

🟥 Testscenario 1: Sessionsterminering vid utloggning

Faktiskt utfall

Användaren omautentiseras då autentiseringscookien är den enda mekanism som autentiserar användaren i nuläget. Backenden håller inte koll på användarens state eller cookiens giltighet utanför om dess access token/refresh token är okej.

Godkänt/Underkänt

UNDERKÄNT med anmärkning.

Sessionsterminering bör utredas - ett alternativ är att lösa SecurityStamps för användare, som sparas i cookien. Om cookiestampen och backendstampen för användaren ej överensstämmer, valideras inte cookien. Backendstampen kan då uppdateras i samband med utloggning. Detta medför dock att alla inloggade sessioner på alla enheter avslutas, något som skulle kunna försämra användarupplevelsen.

WSTG-SESS-07

🟨 Testscenario 1: Sessionsterminering vid inaktivitet

Faktiskt utfall
Användaren loggas ut från applikationen.

Godkänt/Underkänt
GODKÄNT med anmärkning

Inget automatiskt omautentiseringsförsök görs i nuläget.

Så länge cookiens refresh token är giltig, kommer en gammal cookie kunna förnyas. Krav på refresh tokens livstid kan ställas i Okta, och även då att den invalideras om den inte använts på godtycklig mängd tid. Detta är å andra sidan inte nödvändigt om alternativet med SecurityStamps används från anmärkningen i WSTG-SESS-06 testscenario 1.

WSTG-SESS-08

🟩 Testscenario 1: Registrera användare session puzzle

Faktiskt utfall
Det går inte att få tillgång till webadmins konto genom att försöka registrera som denna samtidigt som användaren också har en egen inloggad session.

Godkänt/Underkänt
GODKÄNT

🟩 Testscenario 2: Ändra lösenord session puzzle

Faktiskt utfall
Det går inte att få tillgång till webadmins konto genom att försöka återställa dennes lösenord.

Godkänt/Underkänt
GODKÄNT

🟩 Testscenario 3: Sök upp formulär där en användare kan fylla i användarinformation.

Faktiskt utfall
Det finns ännu inte forms där känslig användardata kan matas in. Det är dock av intresse säkerhetsmässigt att kontrollera eventuella framtida forms så att webbapplikationen inte blir osäker mot session puzzle-attacker. Mer info angående detta kan hittas i OWASPs testguide för testid WSTG-SESS-08

Godkänt/Underkänt
GODKÄNT

WSTG-SESS-09

🟨 Testscenario 1: Omdirigering från HTTP till HTTPS

Faktiskt utfall
Som förväntat kan inga omdirigeringar göras, då IIS-servern är konfigurerad att påtvinga HTTPS för alla förfrågningar.

Godkänt/Underkänt
GODKÄNT med anmärkning

Detta är ett konfigurationstest - webbapplikationen kan hostas med en IIS-konfiguration som inte påtvingar HTTPS. Vid uppsättning av webbapplikationen är det viktigt att detta konfigureras rätt.

WSTG-CRYP-01

Testscenario 1: Säkra Certifikat
Faktiskt utfall
Servercertifikatet som används under utveckling täcker alla krav förutom nr 5 just nu. Detta är dock accepterbart eftersom ett nytt certifikat förväntas införskaffas när webbapplikationen väl sätts i bruk.

Godkänt/underkänt
GODKÄNT med anmärkning

Vid lansering av webbapplikation är det viktigt att detta test utförs då det är ett konfigurationsspecifikt test.

Testscenario 2: Kontrollera kryptering i Nmap
Faktiskt utfall
"TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C" hittas som möjlig cipher suite och TLS 1.0 och TLS 1.1 är möjliga TLS versioner som tillåts

Godkänt/underkänt
UNDERKÄNT med anmärkning

Testet underkänns då både förlegade TLS-versioner och förlegade cipher suites tillåts. När webbapplikationen sätts i bruk bör ovan nämnda cipher suite och TLS-versioner ej tillåtas.

Testscenario 3: Påtvinga HTTP-protokoll
Faktiskt utfall
Det går inte att komma åt sidan via HTTP.

Godkänt/Underkänt
GODKÄNT

WSTG-CRYP-02

Testscenario 1: Padding oracle
Faktiskt utfall
Båda relevanta konfigurationsinställningar var närvarande och var satta till rätt värde. Ingen risk för padding.

Godkänt/Underkänt
GODKÄNT

WSTG-CRYP-03

🟩 Testscenario 1: Lösenord eller krypteringsnycklar i klasser eller konfig-filer.
Faktiskt utfall
Den enda referensen till lösenord och användarnamn är den i web.config till Microsoft SQL-servern. Detta bedöms dock vara säkert eftersom path traversal inte är möjligt för den valda applikationen vilket betyder att denna fil aldrig under några omständigheter kommer vara synlig för frontend.

Godkänt/underkänt
GODKÄNT

🟩 Testscenario 2: Secure session cookie
Faktiskt utfall
Samtliga cookies är av typen Secure.

Godkänt/Underkänt
Godkänt.

WSTG-CRYP-04

🟩 Testscenario 1: Säkra krypteringsmetoder i källkoden
Faktiskt utfall
Inga osäkra metoder hittas förutom användandet av MD5 för att generera ett GUID vilket har kända hash-kollisioner. Detta anses dock av företaget som acceptabelt i nuläget under utveckling men kan komma att ändras i framtiden närmare lansering. Krypteringen som görs av OWIN för t.ex autentiseringscookies görs med maskinnyckel, och kan i IIS konfigureras att använda AES.

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-01

🟩 Testscenario 1: Källkodsgranskning för DOM-based XSS
Faktiskt utfall
Sidan är inte utsatt för DOM XSS i nuläget eftersom inga farliga HTML-metoder hittas där en hackare kan injicera scripts. Webbapplikationens dynamiska uppbyggnad hindrar också till stor del denna typ av attack.

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-02

🟩 Se testspecifikation. Godkänt.

WSTG-CLNT-03

🟨 Testscenario 1: JavaScript-funktioner som är sårbara för HTML-injektioner
Faktiskt utfall
Kodgranskning upptäckte ingen användning av sårbara HTML-metoder.

Godkänt/Underkänt
GODKÄNT med anmärkning

Detta är inte ett uttömmande test för HTML-injektion. Vid vidare utredning, rådslå med OWASP testguide.

🟩 Testscenario 2: HTML-injektion i URL-queryparametrar.
Faktiskt utfall
Sidan godtar inte den injicerade HTML-koden.

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-04

🟩 Testscenario 1: Omdirigering med window.location-objektet.
Faktiskt utfall
Inga omdirigeringar med window.location-objektet kan hittas.

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-05

🟩 Testscenario 1: CSS injection med Nmap.
Faktiskt utfall
Inga sårbarheter rapporteras av Nmap.

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-06

🟩 Testscenario 1: Identifiering av attackytor för resursmanipulation i frontend
Faktiskt utfall

Som väntat. Inga attackytor hittas i nuläget

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-07

🟩 Testscenario 1: Säker användning av CORS
Faktiskt utfall
Ingen säkerhetsbrist hittas där känslig data skickas när CORS-huvudet är satt till en asterisk (wildcard).

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-08

🟩 Testscenario 1: Användning av sårbara ActionScript-funktioner
Faktiskt utfall
Inga ActionScript-referenser hittas. Referenser till htmlText finns som parameternamn, men ej som ActionScript-funktion.

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-09

🟨 Testscenario 1: Penetrationstest med clickjacking
Faktiskt utfall
Clickjackattacken misslyckas då skyddande X-Frame-Options används.

Godkänt/Underkänt
GODKÄNT med anmärkning

Svarshuvudet X-FRAME-OPTIONS är inte kompatibelt med äldre webbläsare (se OWASPs testsida för detta id). Därtill kan en webbproxy ta bort svarshuvudet och nullifiera skyddet. Moderna webbläsare som Chrome varnar dock när en proxy kontrollerar ens anslutning.

Rekommenderade värdet för X-FRAME-OPTIONS är enligt OWASP "DENY", men då EPiServers CMS behöver förhandsvisa webbsidan i en iframe i redaktörsläget är detta inte aktuellt för applikationen.

WSTG-CLNT-10

🟩 Testscenario 1: Användning av WebSockets.
Faktiskt utfall
WebSockets används i nuläget endast i EPiServer, vilket är utanför arbetets scope. Applikationen i sig använder sig inte av WebSockets.

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-11

Testscenario 1: Användning av Cross Document Messaging
Faktiskt utfall
Referenser till postMessage och addEventListeners(‘message’, …) finns endast i den kompilerade frontenden. Parametrar för postMessage är endast *null*, och den enda referensen till addEventListeners som svarar på ‘message’ sköter ingen datamanipulering och bedöms vara säker.

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-12

Testscenario 1: Känslig data lagrad i webbläsare
Faktiskt utfall
Ingen lagringsmekanism läcker användarkontoinformation i klartext. Den enda mekanism som används är Session Storage, som innehåller harmlös skrollningshistorik för användaren.

Godkänt/Underkänt
GODKÄNT

Testscenario 2: Känslig data lagrad i window-objektet
Faktiskt utfall
Window-objektet innehåller ingen användarinformation.

Godkänt/Underkänt
GODKÄNT

WSTG-CLNT-13

Testscenario 1: Dataläckage i JavaScript- eller JSON-responser.
Faktiskt utfall
Den enda användardatan som avslöjas i JavaScript-responserna är användarens epostadress. DetectDynamicJS rapporterade inga fel.

Godkänt/Underkänt
GODKÄNT med anmärkning:

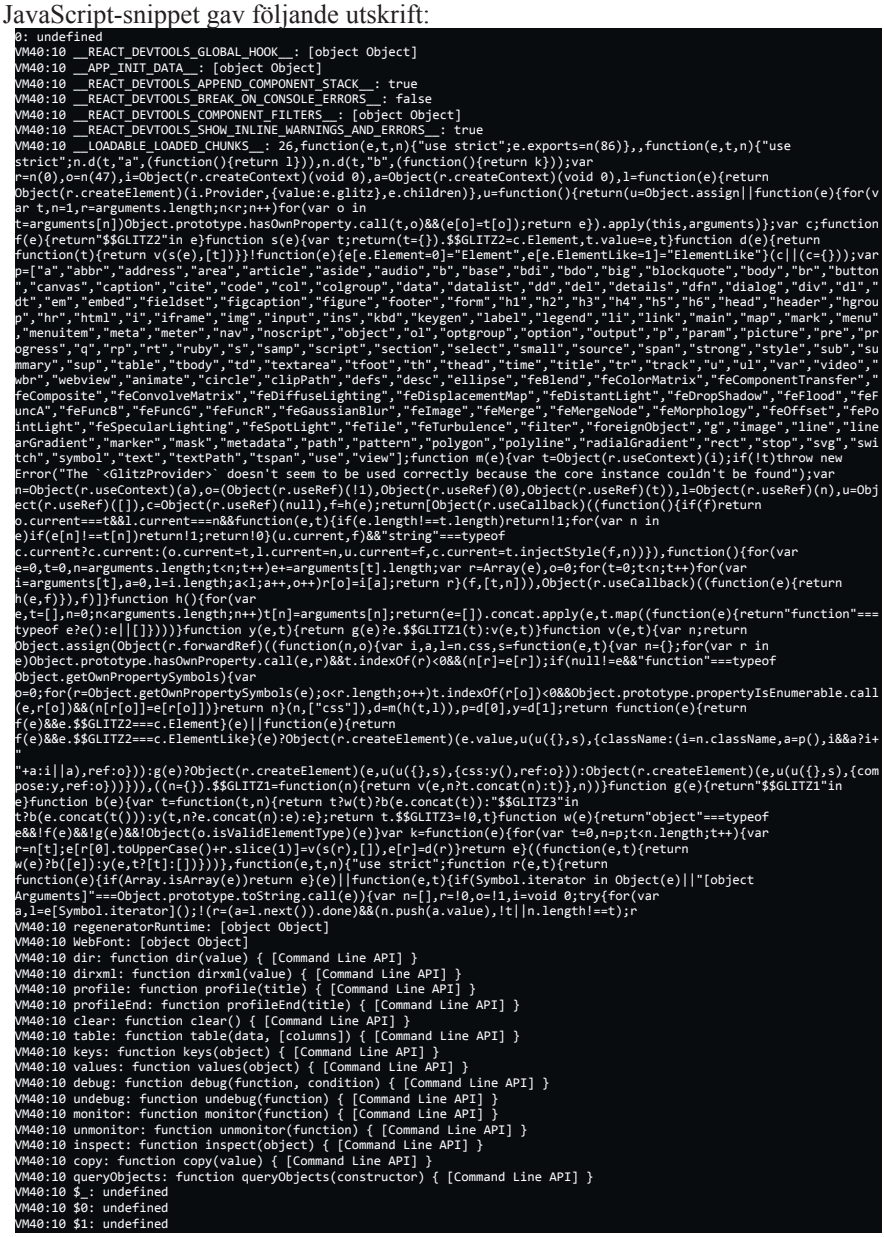
Visserligen avslöjas den inloggade användarens epostadress, men detta bedöms inte vara särskilt känslig information.

Testet har avgränsats till att endast granska JavaScript- och JSON-responser. Andra responsfilformat bör också testas.

Testscenario 2: Dataläckage via globala funktioner.

Faktiskt utfall

Inga funktioner som kan returnera känslig användardata upptäcks. Exekvering av JavaScript-snippet gav följande utskrift:



```
VM40:10 $2: undefined
VM40:10 $3: undefined
VM40:10 $4: undefined
VM40:10 getEventListeners: function getEventListeners(node) { [Command Line API] }
VM40:10 monitorEvents: function monitorEvents(object, [types]) { [Command Line API] }
VM40:10 unmonitorEvents: function unmonitorEvents(object, [types]) { [Command Line API] }
VM40:10 $: function $(selector, [startNode]) { [Command Line API] }
VM40:10 $$: function $$(selector, [startNode]) { [Command Line API] }
VM40:10 $x: function $x(xpath, [startNode]) { [Command Line API] }
```

__APP_INIT_DATA__ innehöll sessionens inloggade användarens e-postadress och guid. Anses vara okänsligt.

Godkänt/Underkänt
GODKÄNT

🟨 Testscenario 3: Dataläckage via CSV with Quotations Theft

Faktiskt utfall
Inga CSV-responser rapporterades.

Godkänt/Underkänt
GODKÄNT med anmärkning.

Då inga responser av typen CSV rapporterades i systemet, kan inte denna attack utföras i nuläget. Dock skulle vidare utveckling kunna införa CSV-responser, vilket öppnar upp möjlighet för attack. För att minimera attackytan bör svarshuvuden ***X-Content-Type-Options: nosniff*** *och* ***Content-Type*** *med charset användas. Därtill förespråkar Takeshi Terada i sitt whitepaper* ***Identifier based XSSI attacks*** *att man inför en eller flera av följande åtgärder för dessa responstyper:*

- *Neka förfrågningar som använder GET*
- *Använd svårgissade parametrar i URL eller i request body.*
- *Använd egenanpassade svarshuvud när innehåll ska hämtas via XMLHttpRequest.*

🟩 Testscenario 4: Dataläckage via Prototype Chaining med nyckelordet this

Faktiskt utfall
Följande scripttaggar skapades:

```
<script src="https://localhost/Dist/build-client/static/js/ProductPage-ProductPage.45460426.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/SearchPage-SearchPage.0fe135d7.chunk.js"></script>
<script
src="https://localhost/Dist/build-client/static/js/OrderConfirmation-OrderConfirmationPage.2c037331.chunk.js"></script
>
<script src="https://localhost/Dist/build-client/static/js/Cart-CartPage.f317b8e8.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/CategoryPage-CategoryPage.63cef8c3.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/MiniCart-MiniCart.c5e7ea80.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/AccountPage-AccountPage.c122ef5a.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/DesktopHeader-DesktopHeader.676c27bf.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/NavigationMenu-NavigationMenu.5f1b9e7f.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/MobileHeader-MobileHeader.85653a72.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/StandardPage-StandardPage.f9a199fe.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/StartPage-StartPage.bef7e033.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/SegmentPage-SegmentPage.e967feb9.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/NotFoundPage-NotFoundPage.e29e7f33.chunk.js"></script>
<script
src="https://localhost/Dist/build-clienthttps://localhost/Dist/build-client/static/js/Blocks-ItemListingBlock-ItemList
ingBlock.19c2ed7b.chunk.js/static/js/CategoryPage-CategoryPage.63cef8c3.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/Models-CartNotificationBanner.d03062ac.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/Models-NotificationBanner.5e5faf30.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/runtime-main.ec04dfb1.js"></script>
<script src="https://localhost/Dist/build-client/static/js/NavigationMenu-UserNavigation.004096d0.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/main.4a2e687c.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/26.7ab44d7b.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/CheckoutPage-CheckoutPage.8c15c134.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/StartPage-StartPage.bef7e033.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/StartPage-StartPage.bef7e033.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/StartPage-StartPage.bef7e033.chunk.js"></script>
```

```
<script src="https://localhost/Dist/build-client/static/js/NotFoundPage-NotFoundPage.e29e7f33.chunk.js"></script>
<script src="https://localhost/Dist/build-client/static/js/ProductPage-ProductPage.45460426.chunk.js"></script>
```

När HTML-koden från steg 3 öppnades i en webbläsare erhölls följande data:

```
Prototype chain values found:
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t
children, dangerouslySetInnerHTML, defaultValue, defaultChecked, innerHTML,
suppressContentEditableWarning, suppressHydrationWarning, style
acceptCharset,accept-charset, className,class, htmlFor,for, httpEquiv,http-equiv
contentEditable, draggable, spellCheck, value
autoReverse, externalResourcesRequired, focusable, preserveAlpha
allowFullScreen, async, autoFocus, autoPlay, controls, default, defer, disabled,
disablePictureInPicture, formNoValidate, hidden, loop, noModule, noValidate, open, playsInline,
readOnly, required, reversed, scoped, seamless, itemScope
checked, multiple, muted, selected
capture, download
cols, rows, size, span
rowSpan, start
accent-height, alignment-baseline, arabic-form, baseline-shift, cap-height, clip-path, clip-rule,
color-interpolation, color-interpolation-filters, color-profile, color-rendering,
dominant-baseline, enable-background, fill-opacity, fill-rule, flood-color, flood-opacity,
font-family, font-size, font-size-adjust, font-stretch, font-style, font-variant, font-weight,
glyph-name, glyph-orientation-horizontal, glyph-orientation-vertical, horiz-adv-x, horiz-origin-x,
image-rendering, letter-spacing, lighting-color, marker-end, marker-mid, marker-start,
overline-position, overline-thickness, paint-order, panose-1, pointer-events, rendering-intent,
shape-rendering, stop-color, stop-opacity, strikethrough-position, strikethrough-thickness,
stroke-dasharray, stroke-dashoffset, stroke-linecap, stroke-linejoin, stroke-miterlimit,
stroke-opacity, stroke-width, text-anchor, text-decoration, text-rendering, underline-position,
underline-thickness, unicode-bidi, unicode-range, units-per-em, v-alphabetic, v-hanging,
v-ideographic, v-mathematical, vector-effect, vert-adv-y, vert-origin-x, vert-origin-y,
word-spacing, writing-mode, xmlns:xlink, x-height
xlink:actuate, xlink:arcrole, xlink:role, xlink:show, xlink:title, xlink:type
xml:base, xml:lang, xml:space
tabIndex, crossOrigin
src, href, action, formAction
animationIterationCount, borderImageOutset, borderImageSlice, borderImageWidth, boxFlex,
boxFlexGroup, boxOrdinalGroup, columnCount, columns, flex, flexGrow, flexPositive, flexShrink,
flexNegative, flexOrder, gridArea, gridRow, gridRowEnd, gridRowSpan, gridRowStart, gridColumn,
gridColumnEnd, gridColumnSpan, gridColumnStart, fontWeight, lineClamp, lineHeight, opacity, order,
orphans, tabSize, widows, zIndex, zoom, fillOpacity, floodOpacity, stopOpacity, strokeDasharray,
strokeDashoffset, strokeMiterlimit, strokeOpacity, strokeWidth
next, throw, return
next, throw, return
tau, alpha, beta, gamma, delta, epsilon, zeta, psi, eta, theta, iota, kappa, lambda, lineHeight,
letterSpacing, fontFamily, fontWeight
primary, secondary, secondaryText, white, breadText, heading, modal, hover, placeholder, sale,
black, red, darkGray, gray, lightGray, errorPrimary, errorText, buttonPrimary, buttonLoading,
transparent
zIndex
animation, timings
mediaQuery, mediaQuerySorter
none, tiny, smaller, small, medium, large, larger, huge, great, vast, gigantic, greater, massive,
enormous, screenMaxWidth, contentMaxWidth, maxWidthSmall, contentResultWidth, maxWidthMobile,
heroMaxHeight, modalHeight, mobileHeaderHeight, thumbnailImageHeight, productImageMaxHeight,
productImageMaxHeightMedium, productImageMaxHeightLarge
0, 5, 6, 7, 16, 17, 18, 19, 20, 22, 23
```

Ingen data som innehöll applikationshemligheter eller användarhemligheter kunde identifieras.

Godkänt/Underkänt
GODKÄNT

Printed by Tryckeriet i E-huset, Lund 2021

LUND
UNIVERSITY

Series of Bachelor's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2021-819
http://www.eit.lth.se