

HACKA DIG SJÄLV OCH UPPTÄCK ATTACKER

HACK YOURSELF AND DETECT ATTACKS

ADNAN SORLIJA

Högskoleingenjör: Datateknik och
Mobil IT

JOHAN FRANSEN

Högskoleingenjör: Datateknik och
Mobil IT

Examensarbete VT19
Kandidatexamen 15 Högskolepoäng
Handledare: Magnus Krampell
Examinator: Steve Dahlskog

Abstract

This thesis is based on the idea of hacking your own system before an outside hacker does it to find the system vulnerabilities. This is done with an automated hacking tool that performs penetration tests against the created website. The database technology that is used is the event database Event Store that stores every event that take place against the website. The task of Event Store in this case is to discover the different penetration tests and to store the events and to give indications to the administrator that the website was under attack.

The study is primarily aimed at finding out whether Event Store is advisable to implement with a website where different penetration testing shall be made, and what the advantages and disadvantages are to using Event Store.

Results show that Event Store can be used to identify anomalies against a website during attacks. Intrusions against the website can with great probability be proven with the help of the developed system with Event Store.

Sammanfattning

Denna uppsats bygger på idén om att hacka det egna systemet före en utomstående hackare gör det för att upptäcka systemets läckor. Detta görs med ett automatiserat hackingverktyg som utför penetrationstester mot en utvecklad hemsida. Lagringstekniken som används är en eventdatabas med namnet Event Store som lagrar varje händelse som skedde mot hemsidan. Syftet med Event Store är att upptäcka de olika penetrationstesterna och lagra dess händelser för att sedan ge indikationer till administratören att hemsidan var under attack.

Uppsatsen riktar sig främst på ifall Event Store är lämpligt att implementera tillsammans med en hemsida som blir attackerad med penetrationstester och vilka för- och nackdelar det finns med att använda Event Store.

Resultatet visar att Event Store kan användas för att identifiera anomalier mot en hemsida vid hackingattacker. Med stor sannolikhet kan intrång mot hemsidan bevisas med hjälp utav det utvecklade systemet med Event Store.

Förord

Rapporten skrevs som en kandidatuppsats för programmet ”Högskoleingenjör: Datateknik och Mobil IT” på Malmö Universitet i Malmö. Idén till arbetet kom från företaget Edument AB som har sitt huvudkontor i Helsingborg. Vi vill främst tacka vår handledare Magnus Krampell och vår examinator Steve Dahlskog för all respons angående framställandet av rapporten och allt arbete runt omkring. Vi vill också tacka vår uppdragsgivare Tore Nestenius från Edument AB för all hjälp kring utvecklandet av produkten.

Ordlista

A-NIDS - Förkortning för Anomaly-based network intrusion detection system.

APT - En APT (Advanced Persistent Threat) attack är en attack som kontinuerligt samlar på sig värdefull information och tränger sig in till målsystemet genom att använda sig av nollårbarheter, socialteknik och andra teknologier. En APT attack görs i fyra steg: intrång, sökning, insamling och attack.

Black box-hackare - Penetrerar systemet på ett icke lagligt sätt och enbart för sin egen vinning.

E2E - End-to-end, förknippas oftast med olika system där hela cykeln täcks från start till slut.

False positive events - Händelser som felaktigt klassificerats som attacker.

Hackingverktyg - Möjliggör penetrationstestning och förebygger sårbarheter.

HTTP - HyperText Transfer Protocol, kommunikationsprotokoll som används för att överföra webbsidor.

Penetrationstestning - Hitta sårbarheter genom att penetrera det egna systemet.

R&D - Research & Development.

SSL - Secure socket layer används för att öka säkerheten i en kommunikation mellan två system och försvåra avlyssning av trafik.

Streaming - När filer överförs i ett nätverk från en webbplats och att filerna som är video- eller ljudfiler, spelas upp i realtid på en mobiltelefon, surfplatta eller dator [30]. Streaming betyder strömning [31].

White box-hackare - Penetrerar systemet på ett lagligt sätt för att hitta sårbarheter.

Innehållsförteckning

1. Inledning	1
1.1 Bakgrund.....	1
1.2 Förbättring av IT-säkerheten.....	1
1.2.1 “Hacka dig själv”	1
1.3 Event Store.....	2
1.4 Syfte	2
1.5 Forskningsfrågor	2
1.6 Avgränsningar.....	3
2. Teknologisnitt	4
2.1 HTTP.....	4
2.2 ASP .NET MVC-applikation	4
2.3 Asynkron programmering C#	5
2.4 Event Store-teknik	5
2.5 Databas.....	6
2.5.1 SQL	6
2.5.2 NoSQL	6
2.6 Penetrationstestning	6
2.6.1 OWASP Zed Attack Proxy	7
2.6.2 Parameter tampering	7
2.6.3 Format String Attack.....	7
3. Relaterat arbete	9
3.1 Event Stream Database Based Architecture to Detect Network Intrusion	9
3.2 A study of Methodologies used in Intrusion Detection and Prevention Systems (IDPS) ...	9
3.3 Anomaly-based network intrusion detection: Techniques, systems and challenges	10
3.4 Big Data Analysis System Concept for Detecting Unknown Attacks	11
4. Metod	13
4.1 Litteratursökning.....	13
4.1.1 Nyckelord.....	13
4.2 Controlled Experiment.....	13
4.2.1 Syfte med experiment.....	13
4.2.2 Kontroll- och Experimentgrupper	13
4.2.3 Variation och repetition.....	14
4.2.4 Genomförande av experiment	14
4.2.5 Analys och utvärdering	14
5. Material	15
5.1 Övergripande systembeskrivning.....	15

5.1.1 Hackingverktyg	16
5.1.2 Eventdatabasen Event Store	16
5.1.3 HTTP Klient	16
5.1.4 OWIN-projektet	16
5.1.5 Konsollapplikation	16
5.2 Utförande av experiment	17
6. Utförande	18
6.1 Process	18
6.2 Utförandet	18
6.2.1 Iteration 1	22
6.2.2 Iteration 2	24
6.2.3 Iteration 3	28
6.2.4 Iteration 4	31
6.2.5 Iteration 5	33
7. Resultat	34
8. Analys och Diskussion	35
8.1 Begränsningar	35
8.2 Jämförelse med relaterat arbete.....	36
8.2.1 Studie A.....	36
8.2.2 Studie B.....	36
8.2.3 Studie C.....	36
8.2.4 Studie D.....	37
9. Slutsats	38
9.1 Frågeställning.....	38
9.1.1 RQ1: Hur kan man använda hackingverktyget OWASP Zed Attack Proxy mot egna webbplatser och tjänster för att förbättra det egna säkerhetssystemet?.....	38
9.1.2 RQ2: Hur kan man upptäcka att man är under attack eller blivit attackerad?	38
9.1.3 RQ3: Är Event Store lämpligt att använda för att upptäcka anomalier som indikerar på hackingattacker?	38
9.2 Förbättringar och framtida utvecklingsmöjligheter	39
Referenser	40
Appendix A	43
Appendix B	44
SQL-injection	44
Cross-site Scripting	44
Forced browsing	44
Appendix C	45
Appendix D	46

1. Inledning

1.1 Bakgrund

Uppsatsen var ett samarbete mellan Malmö Universitet och uppdragsgivaren Edument AB. Idén till företaget Edument AB kom till då grundarna Acke Salem och Tore Nestenius kände att det “saknades tjänster inom utvecklingssektorn som innefattar både utbildning och mentorskap”. Huvudfokuset i företaget ligger inom IT-säkerhet och utbildningar. Därav valdes ett examensarbete med fördjupning inom IT-säkerhet.

1.2 Förbättring av IT-säkerheten

Säkerheten för hemsidor och webbapplikationer är ett konstant problem på grund av att nya sårbarheter ständigt dyker upp. Ingen tvekan råder om att säkerheten kring webbapplikationer är ett aktuellt område. Webbapplikationers säkerhet för intrång berör speciellt hemsidor som behandlar kundinformation, slutanvändare som registrerar känslig information och personer vars mål är att stjäla kunduppgifter och bankkontouppgifter. Få organisationer vill lämna ut information om sina egna säkerhetsproblem och brister, vilket medför att det idag är svårt att få tillförlitlig information om säkerheten kring webbapplikationer [1].

En rapport togs fram 2014 med syftet att visa vilka sätt som är vanligast för att förbättra just IT-säkerheten. Högst upp på listan dyker främst antivirusprogram, byten av lösenord och variation av lösenord för olika konton upp. Längre ner i listan dyker “Hacka dig själv”-begreppet och den mänskliga faktorn upp [2]. Genom att begreppet “Hacka dig själv” dyker upp på listan så behandlar uppsatsen ett av de vanligaste sätten för att förbättra IT-säkerheten.

1.2.1 “Hacka dig själv”

Troy Hunt och Jeremiah Grossman [3] argumenterar för att företagen idag kan förbättra IT-säkerheten genom att försöka hacka det egna systemet innan det egna systemet blir hackat av utomstående. Syftet är att systemets administratör får ny insikt och kunskap i hur systemet bättre kan skyddas mot dagens hot. Företag och privatpersoner borde enligt Troy och Jeremiah utföra penetrationstestning som innebär att utföra attacker på det egna systemet. Tyvärr är nuvarande läge omvänt för de flesta företag och privatpersoner där de istället väntar på att något ska inträffa innan åtgärder tas. Utifrån ett utvecklarperspektiv så prioriteras oftast funktionaliteten i första hand, medan säkerheten ignoreras, vilket möjliggör för utomstående att identifiera och utnyttja misstagen [3].

Många företag använder hackingverktyg för att testa egna systemet mot sårbarheter. Exempel på hackingverktyg är verktyg är Kali Linux [4], Metasploit [5] och OWASP Zed Attack Proxy [6].

Syftet med hackingverktyg är att de enbart används i utbildnings- och testsyfte och får endast användas mot miljöer som brukaren utför attacker mot, annars finns risken för juridiska påföljder. Användandet av hackingverktyg i utbildnings- och testsyfte utförs av s.k. white hat hackers som utför penetrationshacking på ett lagligt sätt [7].

Det är viktigt är att skilja mellan så kallade white hat hackers och black hat hackers. White hat-hackare är personer som hyrs in av företag och har tillstånd att utföra attacker mot företagens system. Syftet med attackerna är att penetrera systemet och hitta svagheter för att företaget eller privatpersonen sedan åtgärdar bristerna [7].

Black hat-hackaren skiljer sig mot white hat-hackaren. Black hat-hackaren utför attacker för att erhålla egen finansiell vinning eller på grund av olika orsaker så utför hackaren attacker mot ett system och utnyttjar systemets brister [8].

Även om problemet med hacking är adresserat och är en stor samhällsfråga i dagsläget så är gapet signifikant mellan antalet hackingincidenter och antalet säkerhetslösningar [9]. Alternativa lösningar kan nyttjas för att minska gapet, förutsatt att administratören ligger ett steg före hela tiden med en strategi för att försvåra hackingförsöken och en korrekt åtgärdsplan om attack har skett [9].

1.3 Event Store

För att möjliggöra spårning eller upptäckande av en potentiell hackingattack som sker mot en hemsida eller en webserver [12] så kan trafiken exempelvis lagras i en databas. Event Store är en databas av typen eventdatabas och i uppsatsen har det har valts att göra fördjupning inom Event Store.

Eventdatabasen Event Store används för att registrera och lagra händelser som inträffar på hemsidan i det utvecklade systemet. Med händelser så menas alla HTTP-förfrågningar som skickas till hemsidan [10]. Events betyder händelser som har skett medan stream beskrivs som en ström som innehåller en sekvens av länkar till de events som har skapats av Event Store [11].

1.4 Syfte

Syftet med uppsatsen var att undersöka IT-säkerhet för webbsidor med hjälp av hackingverktyg och att undersöka ifall den tekniska lösningen med Event Store möjliggjorde att hacka dig själv och upptäcka attacker.

1.5 Forskningsfrågor

Frågor som uppsatsen ska utreda är:

- 1.5.1** RQ1: Hur kan man använda hackingverktyget OWASP Zed Attack Proxy mot egna webbplatser och tjänster för att förbättra det egna säkerhetssystemet?
- 1.5.2** RQ2: Hur kan man upptäcka att man är under attack eller blivit attackerad?
- 1.5.3** RQ3: Är Event Store lämpligt att använda för att upptäcka anomalier som indikerar på hackingattacker?

1.6 Avgränsningar

Efter diskussioner med uppdragsgivaren så togs ett gemensamt beslut att prioritera bort en del forskningsfrågor då större fokus var att besvara frågorna i delkapitel 1.5.

Forskningsfrågor som prioriterats bort är:

- Vilka sårbara webbapplikationer finns som man kan använda för att träna på att hacka? - Frågan prioriterades bort för att fokus riktades på den utvecklade hemsidan istället för att fokusera på vilka andra alternativ det fanns.
- En fullständig steg-för-steg guide gällande hackingdelen och Event Store. - Efter diskussion med handledaren så ansågs undersökningen vara viktigare ifall den tekniska lösningen med Event Store möjliggjorde för idén med att kunna hacka dig själv och upptäcka attacker än att ta fram en steg-för-steg guide. Därför prioriterades guiden bort.
- Visualisera trafiken vid Event Store. - Visualiseringen innebar att utveckla en dashboard som skulle visa datan från Event Store i olika grafer etc. Dashboarden prioriterades bort för att framtagandet av dashboarden ansågs vara ett eget examensarbete storleksmässigt.
- Använda flera hackingverktyg – Uppsatsen skulle ha använt flera hackingverktyg mot utvecklade systemet men efter diskussion tillsammans med uppdragsgivaren valdes bara ett hackingverktyg med motivering att få ut ett så bra resultat som möjligt med ett hackingverktyg istället för halvbra med flertalet. Motiveringen till varför ZAP valdes före de andra hackingverktygen var för att uppdragsgivaren rekommenderade verktyget.
- Vilka alternativ finns för att upptäcka attacker? - Undersökning av andra alternativ än det utvecklade systemet med Event Store prioriterades bort. Efter diskussion med uppdragsgivaren togs beslutet att enbart fokusera på det utvecklade systemet med Event Store för att få ut ett så bra resultat som möjligt.

2. Teknologiansnitt

Nedan beskrivs kortfattat olika teknologier som används inom uppsatsen.

2.1 HTTP

HTTP är ett internetprotokoll och betyder Hypertext Transfer Protocol. HTTP-protokollet används för att etablera kommunikation mellan en server och en klient och för att hantera förfrågningar och svar mellan dem. Exempel på en server är en applikation i datorn som agerar värd för en hemsida och ett exempel på en klient är en webbläsare som kommer åt en hemsida [13].

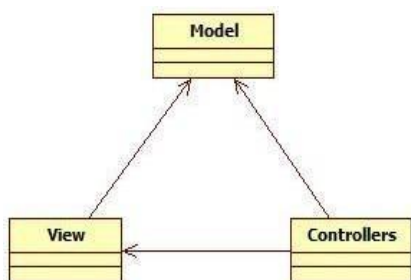
Två vanliga typer av HTTP-förfrågningar är POST och GET. En POST-förfrågan skickar data till en särskild källa där data sedan ska bli behandlad. En GET-förfrågan frågar efter data från en särskild källa [13].

2.2 ASP .NET MVC-applikation

En ASP .NET MVC-applikation är en applikation som bygger på MVC-arkitekturen. Förkortningen MVC står för Model-View-Controller och det är de tre olika komponenter som arkitekturen grundar sig på [26]. Sambandet mellan komponenterna beskrivs nedan.

All datarelaterad logik som en administratör arbetar med motsvaras av Model-komponenten. Datarelaterade logiken representeras av all data som överförs mellan View- och Controller-komponenterna. Exempel på hur Model-komponenten används är när en person hämtar ut kundinformation från en databas och uppdaterar och manipulerar datan innan datan sparas tillbaka. All logik för användargränssnittet av applikationen hanteras av View-komponenten [26]. Exempelvis kan View-komponenten representeras av ett användargränssnitt som innehåller komponenter som dropdown-tabeller, boxar, checkboxar etc. Mellan Model- och View-komponenterna så fungerar tredje delen, Controller-komponenten, som ett gränssnitt. Controller-komponenten används för att exempelvis hantera inkommande förfrågningar till applikationen eller manipulera data i applikationen med hjälp av Model-komponenten. Exempel på hur en Controller-komponent kan användas i ett system är när Controller-komponenten hanterar alla inmatningar och interaktioner från View-komponenten och genom inmatningarna och interaktionerna så uppdaterar och manipulerar Controller-komponenten data i en databas med Model-komponenten [26].

Figur 1 visar sambandet mellan komponenterna.



Figur 1, MVC-arkitektur.

2.3 Asynkron programmering C#

Asynkron programmering i C# introducerades som ett förenklat tillvägagångssätt i C# version 5, vilket har inneburit att det tillkommit asynkron support för .NET Core, Windows Runtime och .NET Framework 4.5 eller högre. För webbåtkomst är asynkron programmering väsentligt eftersom webbåtkomst är en aktivitet som blockerar andra händelser. Exempelvis är asynkron programmering bra när åtkomst behövs till en webbsur. En asynkron process kan parallellt hantera flertal arbeten till skillnad från en synkron process [27].

2.4 Event Store-teknik

Event Store är en open source-databas som består av Complex Event Processing (CEP) i JavaScript [14]. Vid lagring av information till databaser så används CEP till att först utfråga om data innan data lagras eller i fall då data inte lagras till en databas. CEP analyserar och identifierar samband för orsaker mellan händelser i realtid [15]. Detta möjliggör för användare vid olika scenarion att proaktivt vidta åtgärder [42]. CEP ger insikt om vad som händer i ett system genom att kontinuerligt matcha olika händelser (events) mot ett specifikt mönster och genom det kan en användare av ett system agera mot händelser när händelserna inträffar [15].

CEP tillämpas inom exempelvis övervakning av verksamhetsaktiviteter. För övervakning av verksamhetsaktivitet så identifierar CEP möjligheter och problem i tidiga skeden vid övervakning av kritiska resurser och affärsprocesser. CEP tillämpas också vid sensornätverk som övervakar industrianläggningar vid mätning av tex. temperatur och rökintensitet. CEP tillämpas även vid marknadsdata för att hantera råvaru- och lagerpriser [15].

CEP används för att uppfylla kraven att latens ska vara låg och att det ska vara en hög mängd av intrångshändelser per sekund. Med latens menas tiden mellan att en händelse har anlänt och ögonblicket då händelsen blir hanterad. Tiden förväntas vara mindre än några millisekunder men tiden kan ibland vara mindre än en millisekund. Gällande mängden intrångshändelser per sekund så kan mängden vara hundratals till några tusentals händelser per sekund [15].

Mer information om Event Store finns i Appendix A.

2.5 Databas

Skillnader mellan SQL och NoSQL beskrivs nedan.

2.5.1 SQL

SQL står för Structured Query Language och används med huvudsyftet att sköta kommunikationen gentemot en databas och anses vara standardspråket för relationsdata hanteringssystemet enligt ANSI (American National Standards Institute). För att kunna använda en databas med att hämta eller uppdatera data i en databas så används SQL-satser. Några av de vanligaste SQL-satserna är "Select", "Insert", "Update", "Delete", "Create" och "Drop" vilket innefattar mycket som behövs för att använda en SQL-databas [17].

2.5.2 NoSQL

NoSQL som Event Store använder är en förkortning av "Not only SQL" och innebär att en databas inte enbart är baserad på SQL eller inte alls är baserad på SQL, utan även på något annat fråge- och datastrukturspråk. Lättaste sättet att bedöma att en databas är en NoSQL-databas är ifall databasen inte baserar sig på traditionella relationsdatabas-management-system-strukturen (RDBMS) [16]. Ett RDBMS är ett system som består av funktioner och samlingar av program som tillåter användare att administrera, interagera, skapa och uppdatera en relationsdatabas. Språket SQL används av de flesta RDBMS för att komma åt en databas [43].

NoSQL-databaser används för att de ger en högre prestanda och tillgänglighet än relationsdatabaser och har en enklare skalbarhet av datan. NoSQL-databaser är oftast optimerade till en viss sorts data eller frågor tex. Neo4j för graph-frågor och Event Store för events [16].

2.6 Penetrationstestning

Penetrationstestning innebär att datorsystem testas för att upptäcka sårbarheter som hackare skulle kunna utnyttja [18]. Penetrationstester exekveras med olika hackingverktyg som exekverar attacker mot exempelvis en hemsida eller en webbserver. Exempel på hackingverktyg är OWASP Zed Attack Proxy [6].

2.6.1 OWASP Zed Attack Proxy

OWASP Zed Attack Proxy – även kallat ZAP är ett hackingverktyg som används för penetrationstestning, se Figur 2, ZAPStart. ZAP erbjuder möjligheten att automatiskt finna sårbarheter i en webbsida genom att exekvera vanliga attacker mot systemet. Bristerna grupperas sedan i risknivåer: High, Medium, Low, Informational och False Positive [6].



Figur 2, ZAPStart.

Genom QuickStart exekveras olika typer av attacker (Cross Site Scripting, SQL Injection, Directory Browsing, Format String Attack och Parameter Tampering). Några av attackerna som utförs beskrivs nedan men även övriga attacker listas i Appendix B.

2.6.2 Parameter tampering

Parameter tampering indikerar brist på felhantering och potentiella områden och som följd utnyttjas systemet av utomstående i form av cookies och gömda formulärtabeller [28].

2.6.3 Format String Attack

Format String Attack inträffar när en applikation utvärderar en inskickad datasträng mot applikationen som ett kommando. Exempel på kommando är: `printf("The magic number is: %d\n", 1911);`. Kommandot utgörs av flera olika komponenter [29].

En komponent är en formatfunktion som är en ANSIC-konverteringsfunktion som tex. `printf` eller `fprintf` [29]. ANSIC är en samling standarder som används för programspråket C som är publicerade av American National Standards Institute (ANSI) [40]. Formatfunktionen genererar fram en sträng genom en konvertering av en variabel inom det använda programmeringsspråket [29]. Formatfunktionen kan exempelvis se ut som funktionen “`printf`” i kommandoexemplet.

En annan komponent är en formatsträng och det är en typ av ASCII-sträng (ASCIIZ) som innehåller formatparametrar och text och är argumentet till den använda formatfunktionen [29]. ASCII är en förkortning av American Standard Code for Information Interchange [41]. För att representera siffror, bokstäver och andra tecken i en dator så är ASCII det vanligaste systemet att använda för detta [41]. Formatsträngen kan exempelvis se ut som texten “The magic number: %d\n” i kommandoexemplet.

Den sista komponenten är formatsträngparametern som definierar formatfunktionens konverteringstyp, exempel är “%x” och “%s” [29]. Formatsträngparametern kan exempelvis se ut som texten “%d” i kommandoexemplet.

3. Relaterat arbete

Nedan följer en sammanfattning av de mest relevanta vetenskapliga artiklar som hittades inom området *Hacka dig själv och upptäck attacker*. Sökningarna skedde mestadels via Google Scholar med generella sökningar till en början som "hacking", "event store", "detection" och "intrusion" för att sedan begränsa sökningarna ytterligare.

3.1 Event Stream Database Based Architecture to Detect Network Intrusion

Kumaran poängterar problemet som uppstår gällande intrångsförsök som sker när privat information urskiljs från offentlig information. Exempelvis när en bank hanterar privat information som personuppgifter gentemot hantering av offentlig information som nyhetsbrev. Problemet kan förhindras genom ett komplett intrångsdetekteringssystem som traditionellt består av ett signaturbaserat system men ett signaturbaserat system räcker inte alltid då nya hot tillkommer som inte kan upptäckas. Kumaran menar på för att ha ett ordentligt intrångsdetekteringssystem så krävs att systemet känner igen normal användning, missbrukande användning och avvikande användning. Att upptäcka missbrukande användning innebär att upptäcka kända skadliga attacker och att upptäcka avvikande användning innebär att upptäcka hittills okända attacker. Genom att kombinera ett system som känner igen de tre olika typerna av användning med en stor mängd nätverkstrafik och varierande nätverkstrafik i en arkitektur så ges en bra komplexitet [19].

Arkitekturen består av en webbplats som överför filer i ett nätverk till en databas. Missbrukande användning i arkitekturen upptäcks genom att jämföra inkommande events mot en inbyggd lista av kända hot. Avvikande användning upptäcks genom maskininlärning som känner igen mönster. Arkitekturen hanteras av en *data splitter* som delar upp inkommande events till ett eller flera tydliga events som kännetecknas av specifika egenskaper inom fysiska attribut som enhetstyp, MAC-adress, kommunikationsattribut som protokoll, nätverksegenskaper som bandbredd och paketinnehållet i varje event. Eventen hanteras vidare av en *event labeler* som grupperar events genom olika etiketter, där varje etikett är fördefinierad med olika villkor som krävs för att aktiveras. Ett villkor kan vara att enbart vissa IP-adresser är tillåtna medan ett annat villkor kan vara att bandbredden skiljer sig alltför mycket från tidigare events [19].

Lösningen möjliggör för administratören att själv definiera egna villkor och upptäcka attacker i realtid genom att kombinera ett intrångsdetekteringssystem som streamar till en databas [19].

3.2 A study of Methodologies used in Intrusion Detection and Prevention Systems (IDPS)

Mudgingwa et al. beskriver vikten av ett intrångsdetekteringssystem för att förhindra attacker genom monitorering, analys och åtgärd. Författarna har valt att rikta in sig på fyra olika detekteringsmetoder som består av avvikelsebaserad, signaturbaserad, tillståndsbaserad och slutligen hybridbaserad metod där de jämför fördelar och nackdelar för de olika metoderna. Författarna beskriver att detekteringssystem fortsätter utvecklas och förnyas jämfört med att metoderna inte utvecklas i samma takt vilket skapar förvirring i nyare system när metoderna integreras istället [20].

Matrisen i Tabell 1 visar skillnader mellan metoderna, där slutsatsen blev att avvikelsebaserad metod är snäppet bättre än signaturbaserad och tillståndsbaserad metod för att detektera nya hot. De flesta intrångsdetekteringssystem använder idag en kombination av alla fyra metoder som kallas för en hybridlösning [20].

Tabell 1, Parameters for evaluating IDPS methodologies [20].

TABLE 1.
Parameters for evaluating IDPS methodologies.

	Anomaly	Signature	Stateful Protocol Analysis	Hybrid
Resistance to Evasion	Medium	Low	Low	High
High accuracy rate	Medium	Medium	Medium	High
Market Share	Medium	High	Medium	Medium
Scalability	Medium	High	High	Medium
Maturity Level	High	High	High	Medium
Overhead on Monitored System	Medium	Low	Low	Medium
Maintenance	Low	Medium	Medium	Medium
Performance	Medium	High	High	Medium
Easy to Configure	No	Yes	Yes	No
Easy to Use	Medium	Low	Low	Low
Protection against New Attacks	High	Low	Medium	High
False Positives	High	Low	Low	Low
False Negatives	High	Medium	Medium	Low

3.3 Anomaly-based network intrusion detection: Techniques, systems and challenges

Teodoro et al. utgår utifrån ett generellt intrångsdetekteringssystem där arkitekturen delas in i fyra olikablock av funktionsmoduler som innehåller event-boxes, database-boxes, analysis-boxes och response-boxes. Varje block har en funktionalitet, alltifrån monitorering av systemet, lagring av events, analysera events och varna vid fientligt beteende.

Jämförs signaturbaserat detekteringssystem och avvikelsetetekteringssystem, så är signaturbaserat detekteringssystem utmärkt för att upptäcka kända attacker men inte så bra på att upptäcka nya attacker. Avvikelsebaserat detekteringssystem har fördelen att upptäcka nya attacker, nackdelen är att det förekommer stora mängder av "false positive events" som indikerar attacker även om så inte är fallet [21].

Författarna har valt att rikta in sig på avvikelседetekteringssystem, även kallat A-NIDS (anomaly-based network intrusion detection system) i rapporten. Författarna gjorde antagandet att fler egenskaper finns för ett avvikelседetekteringssystem jämfört med ett signaturbaserat detekteringssystem. Författarna har brutit ner A-NIDS tekniker till tre kategorier som är statistiskt baserade, kunskapsbaserade och maskininlärningsbaserade där fördelar och nackdelar påvisas med varje teknik. Författarna har även listat A-NIDS system som finns tillgängliga på marknaden idag och hur de skiljer sig åt. Brister med A-NIDS har listats som har låg detekteringsgrad baserat på “false positive events” och stor underhållskostnad då hög datahastighet har använts (Gbps) [21].

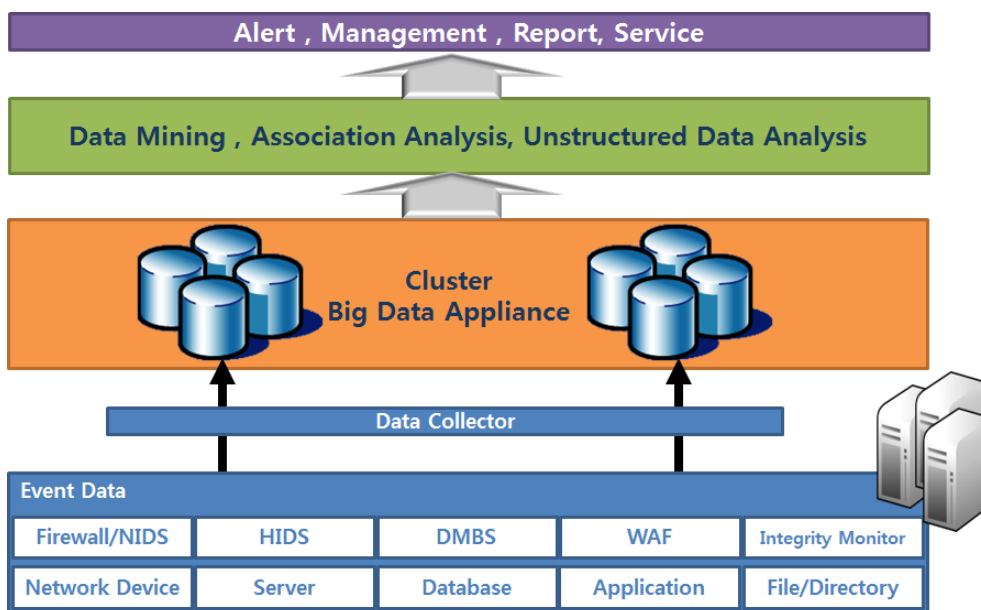
Slutsatsen som följer är att rapporten kan användas som en bra startpunkt i R&D men att bättre och snabbare åtgärder behövs för att klara av den växande mängden attacker [21].

3.4 Big Data Analysis System Concept for Detecting Unknown Attacks

Ahn et al. föreslår en ny modell för att kunna förhindra och detektera okända APT-attacker. En APT-attack inkräktar ett system genom att kontinuerligt samla på sig information som dagens teknologi inte kan detektera. Modellen består av tekniker för “Big Data”-analys som går ut på att reagera på tidigare okända attacker och att kunna detektera framtida attacker genom att extrahera data från olika källor. De försvarsteknologier som är baserade på mönsterigenkänning som motverkar denna typ av attacker är väldigt begränsade och detta gör att detekteringsfrekvensen blir väldigt låg i händelse av både nya och tidigare okända attacker [22].

Författarna fokuserar på fyra tekniker i Big Data-analys: *förutsägelse*, *klassificering*, *relationsregel* och *atypisk data-mining*. Författarna menar för att kunna detektera okända nya attacker så är de fyra teknikerna ett måste. Framtida möjligheter och trender förutspås av den första tekniken *förutsägelse*. En lämplig förutsägelsesteknik är regressionsanalys som t.ex. kan hjälpa forskare att förutsäga angreppsmöjligheter. För att hjälpa en säkerhetsadministratör att välja inriktning för analys och skydd så används *klassificering*. Exempel på klassificering är SVM (Support Vector Machine) och logistisk regressionsanalys. För att upptäcka dolda relationer mellan data så används *relationsregeln*. Genom att analysera process- och användarbeteenden så kan relationsregeln definiera vad som är onormala beteenden. Data som inte kan uttryckas i nummer som tex. ljud, video, bild etc. analyseras av tekniken *atypisk data-mining* [22].

I figur 3 beskriver författarna systemmodellen [22].



Figur3, Bigdata analysis system model [22].

Modellen är indelad i fyrasteg.

Första steget är datainsamlingen som samlar in data från statusinformation och beteende från databaser, anti-virus, nätverk, system samt även eventdata från loggar och brandväggar. Nästa steg är behandlandet av data, där validering av data utförs enbart ifall den insamlade datan uppfyller vissa krav. I näst sista steget analyseras den redan förbehandlade datan. Detta görs genom att använda ostrukturerad dataanalys, associeringsanalys, klassificering och förutsägelse för att bestämma felaktig användning av fil och system, systemstatus, paketintegritet och användarbeteende. Sista steget är resultatet och här larmas administratören och avbryter exekvering av systemet om onormala beteenden upptäcks. Ett hanteringsverktyg och en instrumentpanel för att övervaka resultat i realtid erbjuds också av modellen. Till systemadministratören rapporteras och sammanfattas förutsägelseinformation från det analyserade systemet. Sedan görs också både passiva och automatiska analysmönsteruppdateringar, radering och manipulering av regler och konfigurationsuppdateringar [22].

För att kunna reagera mot tidigare okända internethot föreslås Big Data-systemmodell som slutsats av författarna. Genom förvrängning och kryptering så kringgår nyligen okända attacker befintliga säkerhetslösningar, därför behövs attacktyperna hanteras med nya detekteringsmetoder, i detta fall med en Big Data-systemmodell. Författarna tror att de kan extrahera insamlad statusinformation från olika källor och värdefull information som tidigare varit obefintlig från data med hjälp av Big Data-analys istället för att använda traditionella logganalys eller mönstermatchning för att förutsäga internetattacker. För framtida implementation av APT-förhindrande- och attackdetekteringsystem så förväntar sig författarna att studiens modell med Big Data-analysteknik ska ligga till grund för det [22].

4. Metod

4.1 Litteratursökning

Större delen av insamling av litteratur har gjorts via nätet, Malmö Universitetsbibliotek, Malmö Stadsbibliotek och vid läsandet av böcker om ämnena "*Hacking*" och "*Event Store*". Information om ämnena har hittats separat men desto svårare har det varit att hitta relevanta artiklar av en kombination av ämnena.

4.1.1 Nyckelord

Nyckelord som använts vid sökningar för att besvara forskningsfrågorna: *Hacking, Hacking-tools, Hack yourself, IT Vulnerabilities, IT Security, Event Store, Event Store Security*.

4.2 Controlled Experiment

Metoden som använts som grund för uppsatsen är Controlled Experiment som beskrivs av Easterbrook [23]. Controlled Experiment valdes genom att metoden baseras på att undersöka en testbar hypotes där en eller flera oberoende variabler manipuleras för att se hur det påverkar en eller flera beroende variabler [23]. Metodvalet ansågs vara lämpligt på grund av att uppsatsen baseras på att upptäcka attacker med Event Store vilket beskrivs som en testbar hypotes som metoden Controlled Experiment stödjer. HTTP-förfrågningarna som skickas från ZAP och HTTP Klient jämförs med att en eller flera oberoende variabler manipuleras för metoden. Slutresultatet med konsollapplikationen som innehåller detekteringsregler och genererar en intrångsrapport jämförs med att en eller flera beroende variabler blir påverkade för metoden.

Förutom metodvalet Controlled Experiment fanns alternativet att använda metoden Design Science istället. Efter diskussion med handledaren ansågs Controlled Experiment vara mer lämpligt för uppsatsen.

4.2.1 Syfte med experiment

Genom att hacka dig själv så förhindras framtida attacker då administratören får insikt inom det egna systemet. Syftet var att försöka identifiera anomalier på de attacker som gjordes från hackingverktyget ZAP och bekräfta att en eller flera olika attacker har gjorts och även kunna urskilja attackerna från datan som skickades från HTTP Klient.

4.2.2 Kontroll- och Experimentgrupper

För att kunna få fram ett resultat så har två nästintill identiska system byggts upp under experimentet. Första systemet, HTTP Klient, är ett system med normalt användande. Systemet kallas för en kontrollgrupp och kommer agera som referens på vad som upptäcks vid normalt användande. Andra systemet, ZAP_system, är ett system där intrångsförsök sker från administratören där systemet kallas för experimentgrupp [23].

Skillnaden mellan systemen är att i Figur 4, HttpClient_System används normalt användande samt HTTP Klient för att generera data, medan i Figur 5, ZAP_System använder sig administratören enbart av verktyget ZAP för att generera intrångsförsök och agera en hackare.

Båda experimenten utförs för att urskilja en hackare som försöker göra intrång i ett system gentemot vanligt användande.

4.2.3 Variation och repetition

För att filtrera bort resultat som tillkom av en slump så utförs iterationen av experimentet tio gånger där databasen nollställs vid varje iteration. Se delkapitel 5.2 för mer information.

4.2.4 Genomförande av experiment

Experimentet går ut på att skicka in HTTP-förfrågningar genom ett hackingverktyg och skicka dummy-data i form av HTTP-förfrågningar av en HTTP-klient till en hemsida, för att sedan lagra HTTP-förfrågningarna i en eventdatabas och på ett smidigt sätt få ut en rapport som administratören kan läsa av. Ett fullständigt och övergripligt utförande av experimentet finns i delkapitlet 5.2.

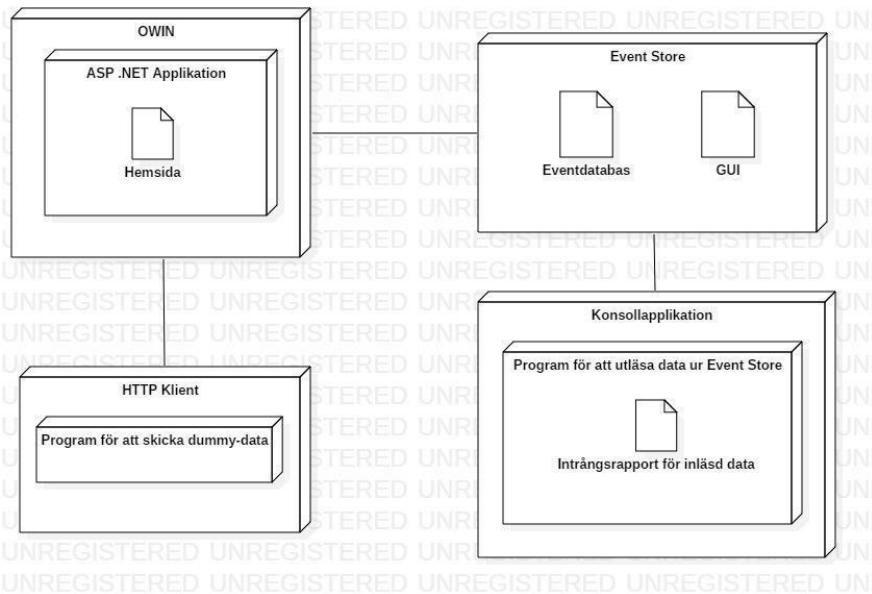
4.2.5 Analys och utvärdering

Analysen av resultatet sker i form av att manuellt gå igenom de events som har registrerats till Event Store från ZAP och HTTP Klient och jämföra eventens data. Utifrån tre påvisade skillnader i eventen skapades detekteringsregler till intrångsrapporten. De tre skillnaderna i eventen presenteras i delkapitel 6.2. Framtagandet av detekteringsreglerna baserades också på den egna kunskapen inom IT-säkerhet. Resterande detekteringsregler som togs fram baserades på teknisk bevisning och vad ZAP presenterade. Vad som menas med teknisk bevisning och vad ZAP presenterade förklaras i delkapitel 6.2. Resultatet presenteras i kapitel 7 Resultat.

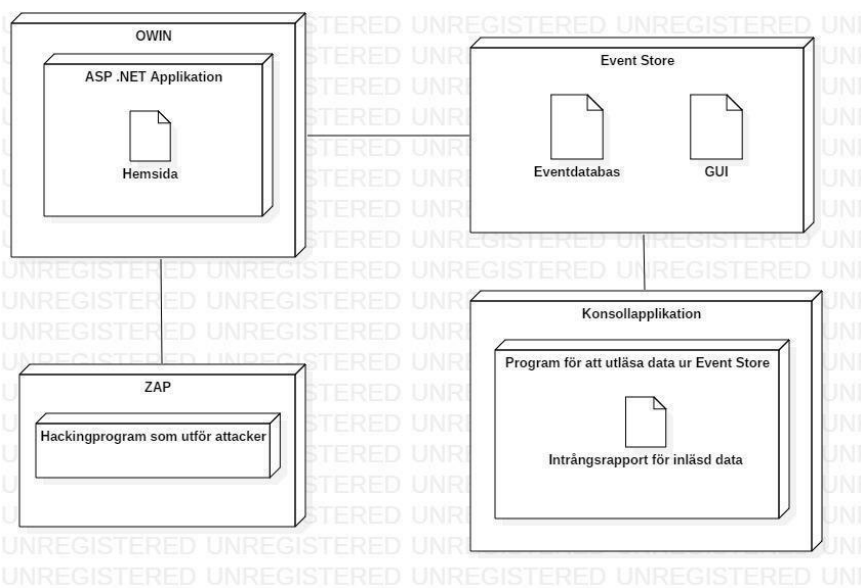
5. Material

5.1 Övergripande systembeskrivning

Det övergripande systemet beskrivs med bilderna nedan där ett av fallen är när hackingverktyget OWASP Zed Attack Proxy används och det andra fallet är när HTTP-klienten användes. Djupare beskrivning av varje del i systemet beskrivs i kommande underkapitel.



Figur 4, HttpClient_System



Figur 5, ZAP_System

Hela systemet har utvecklats och testats lokalt över en och samma dator och inte över ett nätverk. Valet av en och samma dator var för att minska externa faktorerna som kunde påverka resultatet negativt och uppdragsgivaren hade meddelat att systemet skulle användas lokalt på en dator.

5.1.1 Hackingverktyg

ZAP används som hackingverktyg under experimentet. För fler detaljer se, delkapitel 2.5.1.

5.1.2 Eventdatabasen Event Store

För att lagra de olika typerna av HTTP-förfrågningarna som inkommer mot den utvecklade hemsidan så användes det en eventdatabas som heter Event Store.

5.1.3 HTTP Klient

HTTP Klient, ett hjälpprogram som har utvecklats som ett stöd för att utföra E2E-flödet inom systemet genom att sända in flertal HTTP POST-förfrågningar till hemsidan som i sin tur kommunicerar med Event Store där all trafik visualiseras [24]. Målet med förfrågningarna är att generera events till Event Store som skiljer sig mot eventen från ZAP och analysera och se ifall några skillnader kan detekteras. Eventen som skapas i Event Store från HTTP Klient kallas för dummy-data och syftar på trafik i form av HTTP POST-förfrågningar som har skickats till hemsidan med egendefinierad data. För att ta del av utvecklade koden för HTTP Klient, se Appendix C.

5.1.4 OWIN-projektet

I utvecklade systemet med Event Store så användes ett program som heter OWIN-Demo som använder sig av standardgränssnittet OWIN som har programmerats om och anpassats till systemet. OWIN är ett standardgränssnitt med syfte att underlätta kommunikationen mellan .NET webbservrar och webbapplikationer [25]. OWIN-projektet består av en ASP .NET MVC-applikation som utför asynkron programmering vilket möjliggör att flera olika funktioner kan köras parallellt. Applikationen kör tre olika funktioner. En av dem beräknar exekveringstiden, en annan funktion skapar ett ID för varje HTTP-förfrågning som skickas mot applikationen och sista funktionen loggar alla inkomna HTTP-förfrågningar mot applikationen till Event Store. Utöver den asynkrona programmeringen så startar applikationen en hemsida som exekveras i en vald webbläsare.

5.1.5 Konsollapplikation

Konsollapplikation är en applikation som används inom det utvecklade systemet med Event Store, där syftet är att hämta in och analysera data från Event Store. Logiken för konsollapplikationen är uppbyggd genom förutbestämda detekteringsregler som indikerar olika typer av attacker och potentiella attacker. Konsollapplikationen har också detekteringsregler som också indikerar på dummy-data. Under tiden konsollapplikationen exekveras så skrivs all data ut i konsollfönstret och när konsollapplikationen exekverat färdigt så skapas det en intrångsrapport utifrån data som har lästs in från events i Event Store. En utförlig beskrivning av intrångsrapporten finns att läsa i delkapitel 6.2 Utförandet.

5.2 Utförande av experiment

Experimentet bestod i sin helhet av flertal iterationer. För en utförlig beskrivning av de olika iterationerna, finns att läsa i delkapitel 6.2 Utförandet.

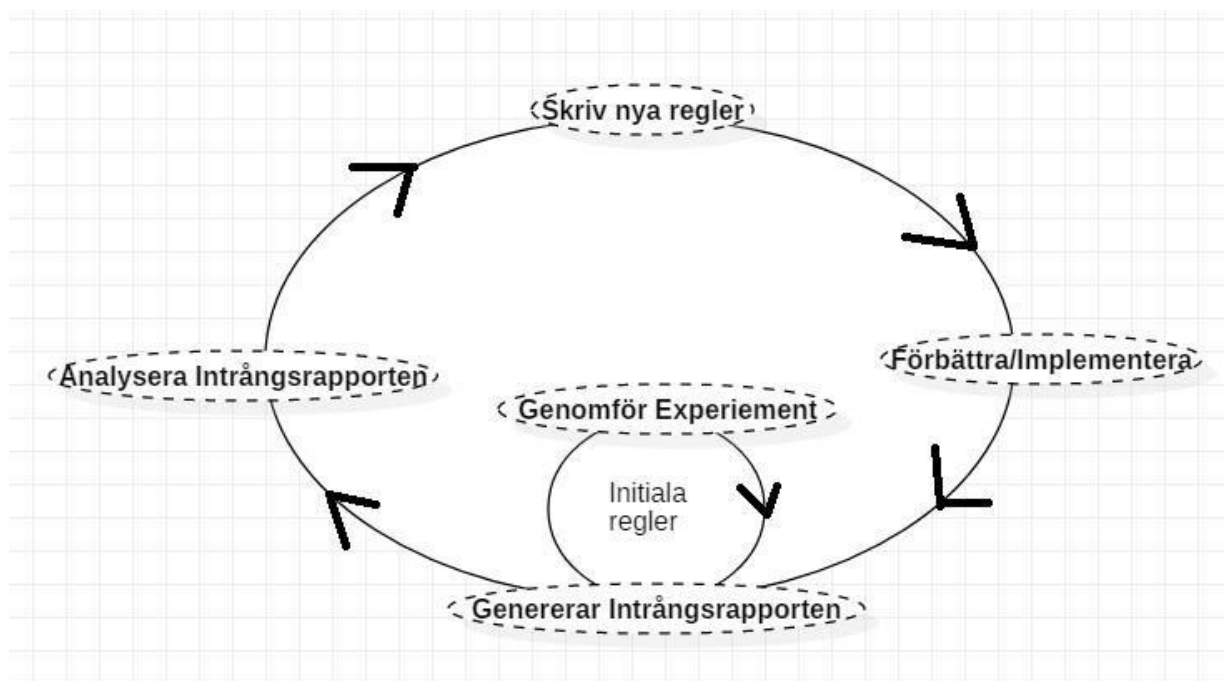
Under varje iteration startades och exekverades Event Store, OWIN-projektet och HTTP Klient i bakgrunden. Där hemsidan kördes som en ASP .NET MVC-applikation via OWIN-projektet och där hemsidan sedan tog emot ett obestämt antal HTTP POST-förfrågningar via HTTP Klient i form av dummy-data. Sedan påbörjades hacking emot hemsidan via ZAP vilket möjliggjorde för konsollapplikationen att generera en intrångsrapport innehållande alla HTTP-förfrågningar hämtade från Event Store. Avslutningsvis gjordes analysen av de events som lagrats i Event Store och detta presenteras i delkapitel 6.2 Utförandet.

Innan en ny iteration påbörjas så startas Event Store, OWIN-projektet och konsollapplikationen om på nytt. Syftet är att nollställa eventdatabasen för att kommande iterationer inte skall påverkas negativt av tidigare lagrad data.

6. Utförande

6.1 Process

Processen för experimentet beskrivs i Figur 6, Experiment-process. Experimentet utfördes i en inre loop vilket resulterade i en intrångsrapport som genereras av konsollapplikationen vid varje iteration. Intrångsrapporten varierar beroende på vilken iteration som utfördes som i sin tur baseras på hur många detekteringsregler som används. I yttre loopen utfördes analys på intrångsrapporten, där nya regler anpassas och implementeras och ett nytt experiment utfördes vilket genererade en ny intrångsrapport. Alla detekteringsregler som användes i de olika iterationerna beskrivs i nästa delkapitel 6.2 Utförandet.



Figur 6, Experiment-process

6.2 Utförandet

Analys av resultatet börjar med att Event Store startas genom dess användargränssnitt via en webbläsare. Därefter väljs menyn Stream Browser → Recently Changed Streams och där väljs en stream som heter "a_test_stream". Vilket är specifika streamen som ASP .NET MVC-applikationen skriver till varje gång en HTTP-förfrågan sker mot hemsidan. Koden inom OWIN-projektet är manuellt konfigurerad så att skrivningar sker emot streamen "a_test_stream" varje gång en HTTP-förfrågan skickas mot hemsidan.

Streamen "a_test_stream" innehåller alla events som skapats genom körningar från HTTP Klient och ZAP. Analysen av ett event i streamen "a_test_stream" sker manuellt genom att öppna ett av flertalet events som finns tillgängliga i streamen "a_test_stream".

Först ut är att öppna ett event i streamen "a_test_stream" för att se över vilken data ett event innehåller som genererats från HTTP Klient. I ett event från HTTP Klient hittades informationen nedan, se Figur 7.

286@a_test_stream

[prev](#) [next](#)

No	Stream
286	a_test_stream

Data

```
{
  "EntryDate": "2019-04-30T19:37:17.3019976+02:00",
  "URL": "http://localhost:16911/Test",
  "IPAddress": "::1",
  "Method": "POST",
  "Headers": [
    "Content-Length:13",
    "Content-Type:application/json; charset=utf-8",
    "Accept:application/json",
    "Expect:100-continue",
    "Host:localhost:16911",
    "X-RequestID:fb6518fb-2335-4c6a-b30a-a698a12b18af"
  ]
}
```

Metadata

```
{"metadata" : Microsoft.Owin.HeaderDictionary}
```

Figur 7, HTTP Klient event

Efteråt öppnades ett event inom samma stream som hade genererats från ZAP för att se vilken data som eventet innehöll. I ett event från ZAP så hittades informationen nedan, se Figur 8.

270@a_test_stream

prev next

No	Stream	Type
270	a_test_stream	RequestData

Data

```
{
  "EntryDate": "2019-04-21T11:57:54.8315488+02:00",
  "URL": "http://localhost:16911/",
  "IPAddress": "::1",
  "Method": "GET",
  "Headers": [
    "Connection:keep-alive",
    "Accept:text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "Accept-Encoding:gzip, deflate",
    "Accept-Language:sv-SE,sv;q=0.8,en-US;q=0.5,en;q=0.3",
    "Host:localhost:16911",
    "User-Agent:Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0",
    "Upgrade-Insecure-Requests:1",
    "X-RequestID:b9ade6d2-d8ab-4008-820e-59683203bf14"
  ]
}
```

Metadata

```
{"metadata" : Microsoft.Owin.HeaderDictionary}
```

Figur 8, ZAP event

Syftet med Figur 7, HTTP Klient event och Figur 8, ZAP event är att ett event öppnades som hade genererats från både HTTP Klient och ZAP för att kunna analysera och hitta likheter och skillnader mellan datan i eventen. Valet av vilka event som analyserades gjordes utifrån det intervall av events som skapades efter att HTTP Klient hade exekverat färdigt, då valdes events inom detta spann. På samma sätt skapades ett intervall av events i samband med körning från ZAP, där events valdes inom detta spann. Intervallet öppnade upp möjligheten för att kunna urskilja events genererade från HTTP Klient respektive ZAP.

Vid analysen för att hitta skillnader och likheter mellan eventen från de olika källorna fångades intresset för parametrarna: URL, Method och Headers. Varför just parametrarna ansågs vara intressanta listas nedan:

- Via parametern URL fås informationen vart HTTP-förfrågan har skickats gentemot hemsidan.
- Via parametern Method fås informationen vilken HTTP-förfrågningsmetod som användes för HTTP-förfrågan.
- Via parametern Headers fås informationen vilka headers eller delparametrar som ändras mellan de olika eventen.

Sammanfattning av likheterna och skillnaderna som hittades efter analysen i alla events från HTTP Klient och ZAP presenteras nedan:

Likheterna som påträffades mellan events:

- Delparametern Content length i Headers fanns i båda eventen.
- Parametern IP-nummer fanns i båda eventen.
- Parametern HTTP-förfrågningsmetod fanns i båda eventen.

Skillnaderna som påträffades mellan events:

I eventen som hade genererats från HTTP Klient förekom parametrar nedan:

- HTTP-förfrågningsmetoden för eventet var POST.
- Värdet för delparametern Content-length var 13.
- IP-nummer-parametern hade värdet: 127.0.0.1.

I eventen som hade genererats från ZAP förekom parametrar nedan:

- Eventet innehöll delparametern Pragma i parametern Headers.
- Eventet innehöll delparametern User-Agent med värdet Mozilla i parametern Headers.
- Värdet för delparametern Content-length var 0.
- IP-nummer-parametern innehöll värdet: ::1.
- HTTP-förfrågningsmetoden för eventet var GET.
- URL-parametern innehöll ord som tex. "ZAP" och "OWASP".

Baserat på tre av skillnaderna inom eventen så skapades det detekteringsregler i konsollapplikationen för att detektera olika potentiella attacker. De tre skillnaderna baserades på att Content-Length hade ett annat värde i eventen från HTTP Klient, eventen från ZAP innehöll parametern Pragma och att URL-parametern för eventen från ZAP innehöll ord som tex. "ZAP" och "OWASP". Motiveringen till detekteringsreglerna presenteras i Tabell 4, Detekteringsregler iteration 1. Beslutet gjordes att inte ta fram detekteringsregler för GET och POST då HTTP-förfrågningstyperna förekommer i alla events. Även för delparametern User-Agent med värdet Mozilla togs beslutet att inte skapa en detekteringsregel då värdet Mozilla är vanligt förekommande och anses därför inte kunna bevisa någon potentiell attack. För IP-parametern skapades det heller ingen detekteringsregel för att IP-adresserna som registrerades från både ZAP och HTTP Klient gäller båda för localhost. IP-adressen till lokala datorn som HTTP Klient och ZAP exekverar på förklaras med ordet localhost [44].

För att undvika att behöva gå igenom varje enskilt event i Event Store eller i konsolfönstret för att bedöma vilka attacker och hur många av varje enskild attack som har skett så fanns behovet att ta fram en lättöverskådlig bild över vilka attacker och potentiella attacker som har skett.

Lösningen till problemet ovan resulterade i att utveckla kod i konsollapplikationen och på så vis få fram en resultatrapport för att ge administratören en lättöverskådlig bild av attackerna och potentiella attackerna. Rapporten namngavs till *Intrångsrapport för hemsidan*, se Appendix D. Detekteringsreglerna för intrångsrapporten modifieras av administratören i konsollapplikationen.

Detekteringsreglerna som togs fram för att identifiera attacker och potentiella attacker gjordes via flera iterationer av systemet där en analys utfördes i slutet av varje iteration för att granska nuvarande och framtida detekteringsregler.

Iteration 2 togs det fram detekteringsregler som byggdes på teknisk bevisning och vilket det även gjordes delvis i iteration 3. Med teknisk bevisning så menas att information har tagits fram

via internet om de olika attackerna SQL Injection, Format String Attack, Forced Browsing, Parameter Tampering, HTTP Only Site och Cross-Site Scripting som skickas från ZAP. Vilket gjordes för att förstå hur respektive attack kan se ut vid andra användningsområden utanför det utvecklade systemet och jämföra det med informationen i events som har loggats till Event Store för att möjligt bevisa att attackerna från ZAP har skett.

I tabeller för de olika detekteringsreglerna i iteration 2 så listas först en förklaring på hur den tekniska bevisningen bevisar att det listade exemplet skulle kunna vara en attack och därefter listas hur detekteringsregeln fungerar.

I tabeller för de olika detekteringsreglerna i iteration 3 så presenteras en förklaring på vad ZAP listar som attacker och teknisk bevisning hur det listade exemplet skulle kunna vara attack. Sist listas hur detekteringsregeln fungerar.

6.2.1 Iteration 1

Iteration 1 exekverades systemet utan att några detekteringsregler var framtagna vilket genererade en tom intrångsrapport. Efteråt togs detekteringsreglerna fram.

Utöver jämförelsen mellan eventen och de tre skillnaderna beskrivet i delkapitel 6.2 så baseras skapandet av detekteringsreglerna också på den egna kunskapen inom IT-säkerhet. Den egna kunskapen inom IT-säkerhet nämns också som egna antaganden längre fram i rapporten. Detekteringsreglerna som togs fram under iteration ett listas i Tabell 4, Detekteringsregler iteration 1.

Tabell 4, Detekteringsregler iteration 1.

Regel	Beskrivning
1. Content-length är större än 0	<p>Regeln valdes för att det upptäcktes skillnader på Content-length parameterens värde i eventen från ZAP och HTTP Klient. Regeln ansågs vara intressant pga. skillnaden i parameterens värde och skillnaden skulle kunna leda till en potentiell attack.</p> <p>Detekteringsregeln söker efter om värdet för parametern Content-length är större än 0.</p>
2. Parametern Pragma finns med i vissa events	<p>Parametern Pragma fanns endast med i events som genererades av ZAP. På grund av det utvecklades en regel som ansågs var intressant eftersom eventen från HTTP Klient inte innehöll parametern och skillnaden skulle kunna vara en potentiell attack.</p> <p>Detekteringsregeln söker efter om parametern Pragma finns med i ett event.</p>
3. Keyword "ZAP" upptäcks i URL-parametern	<p>Regeln valdes för att ordet ZAP bevisar att eventet har genererats från hackingverktyget ZAP eftersom det innehåller en del av namnet för hackingverktyget och därför skulle den typen av event kunna identifiera en potentiell attack.</p> <p>Detekteringsregeln försöker identifiera ordet "ZAP" i ett events URL.</p>
4. Keyword "OWASP" upptäcks i URL-parametern	<p>Regeln valdes för att ordet OWASP bevisar att eventet har genererats från hackingverktyget ZAP eftersom det innehåller en del av namnet för hackingverktyget och därför skulle den typen av event kunna identifiera en potentiell attack.</p> <p>Detekteringsregeln försöker identifiera ordet "OWASP" i ett events URL.</p>

Efter att detekteringsreglerna var framtagna så var iteration 1 klar och då fanns en grund för detekteringsreglerna. Därmed ansågs iteration ett vara avslutad och nästa iteration påbörjades.

6.2.2 Iteration 2

Under iteration två exekverades systemet på nytt för att undersöka huruvida systemet reagerar för de detekteringsregler som lades till under iteration ett. En intrångsrapport genererades efter exekvering av systemet som resulterade i följande:

Tabell 5, Intrångsrapporten iteration 2.

Intrångsrapporten	ZAP	HTTP Klient
Regeln för potentiella hotet "ZAP" har hittats	39 gånger	0 gånger
Regeln för potentiella hotet "OWASP" har hittats	36 gånger	0 gånger
Regeln för potentiella hotet Content-length är större än 0 har hittats	0 gånger	14 gånger
Regeln för potentiella hotet Parametern Pragma finns med I vissa events har hittats	2511 gånger	0 gånger

Efter exekveringen påbörjades analysen av intrångsrapporten för att säkerställa att nuvarande detekteringsregler behöver förbättras eller tas bort. Detekteringsreglerna inom intrångsrapporten initierades och utföll positivt, därmed utfördes inga ändringar gällande de nuvarande detekteringsreglerna.

I resterande del av iteration två skapades det ytterligare detekteringsregler och regler baserade på teknisk bevisning. Bevisning hittades endast för SQL Injection och Format String Attack medan bevisning saknades för resterande attacker. Baserat på den tekniska bevisningen så gjordes slutsatsen att Forced Browsing inte hittades då attacken förknippas med att utelämna vissa sekvenser inom ett inloggningsflöde, vilket saknas helt och hållet på hemsidan. Parameter Tampering hittades inte då där inte fanns tydliga bevis i loggen som pekade på att ändringar gjorts i parametern vid HTTP-anropen som t.ex. ändring av en checkbox på hemsidan. Cross-site Scripting hittades inte då inga tydliga bevis dyker upp som tyder på att utomstående anrop skett gentemot hemsidan i form av t.ex. formulär, bilder m.m. Efter analysen bestod sista steget inom iteration ett av att ta fram detekteringsreglerna. Detekteringsreglerna som togs fram under iteration två listas i Tabell 6, Detekteringsregler iteration 2.

Tabell 6, Detekteringsregler iteration 2.

Regel	Beskrivning
<p>5. SQL Injection</p>	<p>SQL Injection försökte identifieras i de events som hade sparats ner till Event Store från hemsidan och attacken hittades i några events efter att teknisk research på internet hade gjorts om hur en SQL Injection kan se ut vid en HTTP-förfrågan.</p> <p>Exempel på hur en SQL Injection kan se ut genom HTTP-förfrågan [33]: <code>http://www.example.com/index.php?username=1'%20or%20'1'%20=%20'1&password=1'%20or%20'1'%20=%20'1</code></p> <p>Exempel på hur en SQL Injection kan se ut som är genererad från ZAP: <code>{[http://localhost:16911/Content?query=query'%26cat+%2Fetc%2Fpasswd%26', 3]}</code></p> <p>På OWASPs hemsida så beskrivs det hur SQL Injection kan testas vid exempelvis en HTTP GET-förfrågning. I exemplet 1 under Standard SQL Injection Testing på länken så testas det att skriva till SQL-frågan nedan i en HTTP GET-förfrågan [33]: <code>SELECT * FROM Users WHERE Username='1' OR '1' = '1' AND Password='1' OR '1' = '1'</code>. Ifall SQL-frågan ovan läggs till i en HTTP GET-förfrågan mot adressen <code>http://www.example.com/index.php</code> såg resultatet ut så här: <code>http://www.example.com/index.php?username=1'%20or%20'1'%20=%20'1&password=1'%20or%20'1'%20=%20'1</code></p> <p>I URL-parametern för några event från Event Store så hittades det ord som var likvärdiga med SQL Injection testet på OWASPs hemsida. Ordet “passwd” hittades i URL-parametern för flera event. Ordet “passwd” tyder på att det kan vara en SQL Injection som har gjorts av ZAP. Se exemplet nedan för hur URL-parametern såg ut för ett event i Event Store som innehöll ordet “passwd”. <code>{[http://localhost:16911/Content?query=query'%26cat+%2Fetc%2Fpasswd%26', 3]}</code></p>

	<p>Det som påvisas är att ett event i Event Store har ett värde, "passwd", som är likt värdet "password" för URL:en som används i SQL Injection-testet på OWASPs hemsida.</p> <p>Kan även påvisas att ordet "passwd" i URL-parametern från eventet i Event Store är med som en parameter till huvudparametern "query" som används för utfrågning. Det kan jämföras med länken från SQL Injection testet: http://www.example.com/index.php?username=1'%20or%20'1'%20=%20'1&password=1'%20or%20'1'%20=%20'1 där ordet "password" är med som en parameter i SQL-frågan som används också som utfrågning. I båda GET-förfrågningarna så görs det möjligtvis en utfrågning om password och är ett möjligt bevis på att hittade URL-parametern i Event Store är en SQL Injection attack.</p> <p>Detekteringsregeln som togs fram för att identifiera en SQL Injection attack går ut på att konsollapplikationen letar efter ordet "passwd" i ett eventsURL-parameter.</p>
<p>6. Format String Attack</p>	<p>Format String Attack identifieras i de event som sparats ner till Event Store från hemsidan och attacken identifierades i några events efter att teknisk undersökning på internet hade gjorts om hur Format String Attack kan se ut vid en HTTP-förfrågan.</p> <p>Exempel på hur en Format String Attack kan se ut [34]: http://hostname/cgi-bin/query.cgi?name=john%x.%x.%x&code=45765%x.%x</p> <p>Exempel på hur en Format String Attack kan se ut genererad från ZAP: <pre>{[http://localhost:16911/?query=%24{ %40print(chr(122).chr(97).chr(112).chr(95).chr(116).chr(111).chr(107).chr(101).chr(110))}%5C, 2]}</pre></p> <p>Utifrån hur Format String Attack kan se ut så upptäcktes i flera event i Event Store att eventens URL-parameter innehöll ordet print på formen:</p>

	<pre>{[http://localhost:16911/?query=%24{%40print(chr(122).chr(97).chr(112).chr(95).chr(116).chr(111).chr(107).chr(101).chr(110))}%5C, 2]}</pre> <p>Exemplet ansågs vara likt exemplet i teoriavsnittet i delkapitel 2.6.3 om hur Format String Attack kan se ut med det angivna kommandot:</p> <pre>printf("The magic number is: %d\n", 1911);</pre> <p>där ordet print också finns med.</p> <p>Allt efter ordet printf i exemplet: printf ("The magic number is: %d\n", 1911); skulle kunna motsvaras av allt som kommer efter ordet print i exemplet från URL-parametern i ett event i Event Store:</p> <pre>print(chr(122).chr(97).chr(112).chr(95).chr(116).chr(111).chr(107).chr(101).chr(110))}%5C, 2].</pre> <p>Liknelsen mellan en URL-parameter från ett event i Event Store och hur Format String Attack kan se ut ansågs vara ett tillräckligt bra tekniskt bevis för att bevisa att URL-parametern från ett event i Event Store skulle kunna vara HTTP-förfrågan som är Format String Attack.</p> <p>Detekteringsregeln som togs fram för att identifiera Format String Attack går ut på att konsollapplikationen letar efter ordet "print" i URL:en för ett event.</p>
--	--

Nya detekteringsregler från iteration två lades till i konsollapplikationen, tillsammans med övriga detekteringsregler från föregående iteration. Därmed ansågs iteration två vara avslutad och nästa iteration påbörjades.

6.2.3 Iteration 3

Under iteration tre så exekverades systemet på nytt som i föregående iterationer. Syftet under iteration tre bestod av att exekvera samtliga detekteringsregler inom intrångsrapporten och verifiera resultatet och sedan lägga till ytterligare detekteringsregler. En intrångsrapport genererades efter exekvering av systemet som resulterade i följande:

Tabell 7, Intrångsrapport iteration 3.

Intrångsrapporten	ZAP	HTTP Klient
Regeln för hotet SQL Injection och ordet "passwd" finns med i ett events URL, har hittats	16 gånger	0 gånger
Regeln för hotet Format String Error och ordet "print" finns med i ett events URL, har hittats	19 gånger	0 gånger
Regeln för potentiella hotet "ZAP" har hittats	28 gånger	0 gånger
Regeln för potentiella hotet "OWASP" har hittats	24 gånger	0 gånger
Regeln för potentiella hotet Content-length är större än 0 har hittats	0 gånger	16 gånger
Regeln för potentiella hotet Parametern Pragma finns med I vissa events har hittats	2689 gånger	0 gånger

Efter exekveringen så påbörjades analys av intrångsrapporten för att säkerställa nuvarande detekteringsregler behöver förbättras eller tas bort. Detekteringsreglerna i intrångsrapporten initierades och resulterade positivt, därmed utfördes inga ändringar gällande nuvarande detekteringsreglerna.

I resterande del av iteration tre skapades det ytterligare detekteringsregler. Reglerna baserades på den tekniska bevisningen, den egna kunskapen inom IT-säkerhet och det som presenterades utav ZAP.

ZAP indikerade på att attacken SQL Injection hade skett mot systemet, vilket en detekteringsregel redan hade skapats för från iteration två. Hoten HTTP Only Site och CSRF Tokens Scanner identifierades som höga och medelstora hot i ZAP. Baserat på hoten jämfördes resultatet från ZAP med analys av events för att identifiera var i systemet hoten skett, vilket resulterade i att hoten hittades även i analysen av events. En teknisk bevisning gjordes på egen hand via webben om både HTTP Only Site och CSRF Tokens Scanner för att säkerställa att hoten inom systemet stämmer överens med hur respektive attack kan se ut. Detekteringsreglerna som togs fram under iteration tre listas i Tabell 8, Detekteringsregler iteration 3.

Tabell 8, Detekteringsregler iteration 3.

Regel	Beskrivning
7. HTTP Only Site	<p>Attacken HTTP Only Site utförs genom att administratören försöker anropa en hemsida som saknar SSL (Secure Socket Layer) vilket krypterar en tvåvägskommunikation [37].</p> <p>Attacken identifierades av ZAP efter avslutad körning, varpå en analys av events gjordes för att verifiera var i systemet attacken gjordes. Utöver detta gjordes en teknisk undersökning på webben om hur en HTTP Only Site attack kan se ut, för att säkerhetsställa bevisningen för att en HTTP Only Site attack verkligen skett.</p> <p>Attacken sker genom att administratören anropar HTTPS istället för HTTP. Nuvarande system påvisar en medium risk gällande just HTTP vilket kan utnyttjas i form av en HTTP Only Site attack [35].</p> <p>Exempel på hur en HTTP Only Site attack kan se ut [35]: http://example.com/myaccount</p> <p>Inom ZAP ser risken ut på följande sätt: ZAP attempted to connect via: https://localhost:443/</p> <p>Under analys av events hittades följande värde för parametern URL i två events:</p> <ol style="list-style-type: none"> 1. {[http://localhost:16911/Content?query=HtTp:%2F%2F707681151828994876.owasp.org, 1]} 2. {[http://localhost:16911/Content?query=HtTpS:%2F%2F7076811518289948876.owasp.org, 1]} <p>Jämförs hur en HTTP Only Site attack kan se ut och hur innehållet i ZAP ser ut, så är de snarlika där bägge använder sig av HTTP. Inom URL-parametern från de två evenen ovan så påvisas att en förfrågan skett mot både HTTP och HTTPS och att i exemplet med ZAP så försöker ZAP ansluta mot https://localhost:443/ men misslyckas då hemsidan saknar SSL. Bevisningen anses vara hög för att HTTP Only Site attack kan ha</p>

	<p>skett och att en detekteringsregel tas fram för att detektera liknande attacker i framtiden.</p> <p>Detekteringsregeln som togs fram för att identifiera HTTP Only Site attack går ut på attacken letar efter ordet "HTTPS" i alla events URL-parameter.</p>
<p>8. CSRF Tokens Scanner</p>	<p>Attacken CSRF Tokens Scanner utförs genom att administratören skickar oönskade HTTP-förfrågningar utan slutanvändarens kunskap [36].</p> <p>Attacken identifierades av ZAP efter avslutad körning, varpå en analys av events gjordes för att verifiera var i systemet attacken skedde. En teknisk undersökning gjordes på webben om hur CSRF Tokens Scanner attack kan se ut, detta för att säkerhetsställa bevisningen för att en CSRF Token Scanner attack verkligen skett.</p> <p>Attacken sker genom att administratören använder hemsidan, skriver in hemsidans URL direkt i webbläsaren eller via externa länkar [36]. Nuvarande system påvisar en hög risk gällande just formuläret som används för att utföra en sökning på hemsidan, vilket kan utnyttjas i form av en CSRF attack.</p> <p>Exempel på hur en CSRF kan se ut:</p> <pre><form action='http://tagetWebsite/Authenticate.jsp' method='POST' name='CSRF'></pre> <p>Exempel på hur CSRF ser ut inom ZAP:</p> <pre><form id="tfnewsearch" method="get" action="http://www.google.com"></pre> <p>Under analys av events hittades följande sökning i parametern URL:</p> <pre>{ [http://localhost:16911/Content/?query=http: %2F%2Fwww.google.com%2Fsearch%3Fq%3 DOWASP%2520ZAP, 1]}</pre> <p>Jämförs exemplet på hur en CSRF attack kan se ut och hur innehållet i ZAP ser ut, så är dem snarlika. Undersöks URL-parametern i exemplet ovan från analys av events så påvisas att en googlesökning skett i formuläret på hemsidan i form av en förfrågan via ZAP. Därmed anses bevisningen vara hög för att en CSRF attack kan</p>

	<p>ha skett och en detekteringsregel tas fram för att detektera liknande attacker i framtiden.</p> <p>Detekteringsregel letar efter ordet “search” i ett events URL.</p>
--	--

Denya detekteringsreglerna från nuvarande iteration lades till i konsollapplikationen, tillsammans med övriga detekteringsregler från föregående iteration. Därmed ansågs iteration tre vara avslutad och nästa iteration påbörjades.

6.2.4 Iteration 4

Under fjärde iteration utfördes exekvering av systemet på nytt. Syftet under iteration fyra bestod av att exekvera samtliga detekteringsregler inom intrångsrapporten och verifiera resultatet. En intrångsrapport genererades efter exekvering av systemet som resulterade i följande:

Tabell 9, Intrångsrapport iteration 4.

Intrångsrapporten	ZAP	HTTP Klient
Regeln för hotet HTTP Only Site har hittats	11 gånger	0 gånger
Regeln för hotet Anti CSRF Tokens Scanner har hittats	13 gånger	0 gånger
Regeln för hotet SQL Injection och ordet "passwd" finns med i ett events URL, har hittats	14 gånger	0 gånger
Regeln för hotet Format String Error och ordet "print" finns med i ett events URL, har hittats	21 gånger	0 gånger
Regeln för potentiella hotet “ZAP” har hittats	31 gånger	0 gånger
Regeln för potentiella hotet “OWASP” har hittats	32 gånger	0 gånger
Regeln för potentiella hotet Content-length är större än 0 har hittats	0 gånger	19 gånger
Regeln för potentiella hotet Parametern Pragma finns med I vissa events har hittats	2544 gånger	0 gånger

Efter exekveringen så påbörjades analys av intrångsrapporten för att säkerställa ifall nuvarande detekteringsregler behöver förbättras eller tas bort. Detekteringsreglerna inom intrångsrapporten initierades och resulterade positivt, därmed utfördes inga ändringar gällande nuvarande detekteringsregler.

Under resterande del av iteration fyra så skapades det ytterligare några nya detekteringsregler till konsollapplikationen. De nya detekteringsreglerna handlade om att klassa keywords på kända hot som “Trojan”, “Adware” och “Spam” som attacker. Detekteringsreglerna baserades på den egna kunskapen inom IT-säkerhet och listas i Tabell 10, Detekteringsregler iteration 4.

Tabell 10, Detekteringsregler iteration 4.

Regel	Beskrivning
9. Keyword "Trojan"	<p>Regeln valdes då ordet "Trojan" oftast förknippas i negativ term och ses oftast som ett hot.</p> <p>Detekteringsregeln försöker leta efter ordet "Trojan" i ett events URL-parameter.</p>
10. Keyword "Adware"	<p>Regeln valdes då ordet "Adware" oftast förknippas i negativ term och ses oftast som ett hot.</p> <p>Detekteringsregeln försöker leta efter ordet "Adware" i ett events URL-parameter.</p>
11. Keyword "Spam"	<p>Regeln valdes då ordet "Spam" oftast förknippas i negativ term och ses oftast som ett hot.</p> <p>Detekteringsregeln försöker leta efter ordet "Spam" i ett events URL-parameter.</p>

De nya detekteringsreglerna från nuvarande iteration lades till i konsollapplikationen tillsammans med övriga detekteringsregler från föregående iteration. Därmed ansågs iteration fyra vara avslutad och nästa iteration påbörjades.

6.2.5 Iteration 5

Under femte iterationen utfördes exekveringen av systemet återigen. Tanken under iteration fem bestod av att exekvera samtliga detekteringsregler inom konsollapplikationen och verifiera resultatet. En intrångsrapport genererades efter exekvering av systemet som resulterade i följande:

Tabell 11, Intrångsrapport iteration 5.

Intrångsrapporten	ZAP	HTTP Klient
Regeln för hotet HTTP Only Site har hittats	8 gånger	0 gånger
Regeln för hotet Anti CSRF Tokens Scanner har hittats	18 gånger	0 gånger
Regeln för hotet SQL Injection och ordet "passwd" finns med i ett events URL, har hittats	23 gånger	0 gånger
Regeln för hotet Format String Error och ordet "print" finns med i ett events URL, har hittats	21 gånger	0 gånger
Regeln för potentiella hotet "ZAP" har hittats	39 gånger	0 gånger
Regeln för potentiella hotet "OWASP" har hittats	36 gånger	0 gånger
Regeln för potentiella hotet Content-length är större än 0 har hittats	0 gånger	19 gånger
Regeln för potentiella hotet Parametern Pragma finns med I vissa events har hittats	2511 gånger	0 gånger
Regeln för potentiella hotet "Trojan" har hittats	0 gånger	0 gånger
Regeln för potentiella hotet "Adware" har hittats	0 gånger	0 gånger
Regeln för potentiella hotet "Spam" har hittats	0 gånger	0 gånger

Efter exekveringen så påbörjades analys av intrångsrapporten för att säkerställa ifall nuvarande detekteringsregler behöver förbättras eller tas bort. De flesta detekteringsreglerna inom intrångsrapporten initierades och resulterade positivt, dock stod det klart att detekteringsreglerna för "Trojan", "Adware" och "Spam" inte triggades. Därmed utfördes ändringar i detekteringsreglerna. Detekteringsreglerna "Trojan", "Adware" och "Spam" valdes att avlägsnas från konsollapplikationen då detekteringsreglerna inte tillför någon nytta gentemot systemet.

Under femte iterationen hittades inga förbättringsförslag för detekteringsregler inom konsollapplikationen. Därmed ansågs iteration fem vara avslutad, som resulterade i den sista iterationen då inga fler förbättringsförslag dyker upp efter ett flertal analyser av events.

Då iteration fem är slutgiltiga iterationen så exekveras systemet tio gånger för att utesluta misstankar om att slutgiltiga resultatet tillkommit av en slump. En intrångsrapport genererades efter exekvering av systemet som resulterade i allt som presenteras i följande:

Tabell 12, Intrångsrapport efter tio exekveringar.

Intrångsrapporten	ZAP	HTTP Klient
Regeln för hotet HTTP Only Site har hittats	16 gånger	0 gånger
Regeln för hotet Anti CSRF Tokens Scanner har hittats	24 gånger	0 gånger
Regeln för hotet SQL Injection och ordet "passwd" finns med i ett events URL, har hittats	35 gånger	0 gånger
Regeln för hotet Format String Error och ordet "print" finns med i ett events URL, har hittats	18 gånger	0 gånger
Regeln för potentiella hotet "ZAP" har hittats	45 gånger	0 gånger
Regeln för potentiella hotet "OWASP" har hittats	32 gånger	0 gånger
Regeln för potentiella hotet Content-length är större än 0 har hittats	0 gånger	19 gånger
Regeln för potentiella hotet Parametern Pragma finns med I vissa events har hittats	2622 gånger	0 gånger

Resultatet av den slutgiltiga iterationen med alla likheter och skillnader som hittades mellan eventen från HTTP Klient och ZAP presenteras i nästa kapitel 7 Resultat.

7. Resultat

Genom det utvecklade systemet med Event Store som tagits fram med detekteringsregler så kan attacker med stor sannolikhet upptäckas baserat på teknisk bevisning och potentiella attacker som baserats på antagandet att vara attacker från ZAP i form av HTTP-förfrågningar som har skickats mot den utvecklade hemsidan. Det har även registrerats dummy-data som potentiella attacker från HTTP Klient som fungerade som referensdata gentemot data som skapats från ZAP.

Utifrån alladetekteringsreglerna så blev det slutgiltiga resultatet för intrångsrapporten nedanstående tabell:

Tabell 13, Slutgiltiga intrångsrapporten.

Intrångsrapporten	ZAP	HTTP Klient
Regeln för hotet HTTP Only Site har hittats	17 gånger	0 gånger
Regeln för hotet Anti CSRF Tokens Scanner har hittats	24 gånger	0 gånger
Regeln för hotet SQL Injection och ordet "passwd" finns med i ett events URL, har hittats	36 gånger	0 gånger
Regeln för hotet Format String Error och ordet "print" finns med i ett events URL, har hittats	20 gånger	0 gånger
Regeln för potentiella hotet "ZAP" har hittats	42 gånger	0 gånger
Regeln för potentiella hotet "OWASP" har hittats	36 gånger	0 gånger
Regeln för potentiella hotet Content-length är större än 0 har hittats	0 gånger	19 gånger
Regeln för potentiella hotet Parametern Pragma finns med i vissa events har hittats	2666 gånger	0 gånger

Slutgiltiga resultatet baseras på den sortering som skett mellan teknisk bevisning, den egna kunskapen inom IT-säkerhet och presentation av ZAP, se Tabell 14. Mer detaljerad information om sortering finns i delkapitel 6.2 Utförandet.

Tabell 14, Sortering för intrångsrapporten.

Teknisk bevisning	Den egna kunskapen inom IT-säkerhet	Presentation av ZAP
SQL Injection	Content length större än 0	HTTP Only Site
Format String Error	Parametern pragma finns med i vissa events	Anti CSRF Scanner
	Ordet ZAP har hittats i ett events URL	
	Ordet owasp har hittats i ett events URL	

8. Analys och Diskussion

Tanken med intrångssystemet var att göra ett system säkrare genom att hacka dig själv och på så vis upptäcka hackingattacker med hjälp av en loggning mot Event Store. Upptäcka hackingattacker gjordes genom att logga HTTP-förfrågningar som skickades in till hemsidan för att sedan exempelvis identifiera olika typer av hackingattacker och att på så vis få fram ett slutgiltigt resultat.

Utifrån resultatet från Tabell 13 som genererats så är det tydligt att attackerna SQL Injection, Format String Attack, HTTP Only Site och CSRF som har baserats på teknisk bevisning har hittats ett antal gånger vardera och sammanfattningsvis så visar resultatet att de olika attackerna har med stor sannolikhet kunnat identifieras. Det är även tydligt att de potentiella attackerna som har baserats på det egna antaganden har hittats ett antal gånger. Granskas resultatet från Tabell 13 Slutgiltiga intrångsrapporten så är det tydligt att ett stort antal möjliga hot har identifierats under sista iterationen, hela 2860 potentiella antal hot har identifierats. Av 2860 potentiella hot så tillhör enbart 97 hot som grundar sig på den tekniska bevisningen medan resterande tillhör antaganden gjorda på egna antaganden samt dummy-data utifrån HTTP Klient. Vilket beror på att detekteringsreglerna som är baserade på den tekniska bevisningen endast detekterar en liten andel events av de detekteringsregler som har sparats till EventStore.

De resterande 96.7% potentiella hot som inte utgår från teknisk bevisning baseras på detekteringsregler som är baserade på det egna antaganden och dummy-data från HTTP Klient. Det syns från Tabell 13 att de potentiella attackerna utgör den större delen av intrångsrapporten. Det kan härledas till att potentiella attackerna baserat på egna antaganden och dummy-data dyker i större utsträckning upp än attacker för teknisk bevisning. Detekteringsregeln för parametern Pragma är ett tydligt exempel på en potentiell attack baserad på eget antagande då parametern Pragma har identifierats 2666 gånger. Detekteringsregeln för parametern Pragma fick så högt värde för att parametern Pragma fanns med i events genererade av ZAP. Jämförs detekteringsregeln för parametern Pragma med exempelvis detekteringsregeln för SQL Injection som hade ett betydligt lägre värde så härleds skillnaden till att ordet "passwd" enbart dök upp i de events från ZAP som innehöll ordet "passwd" i URL-parametern.

8.1 Begränsningar

För att möjliggöra den slutgiltiga intrångsrapporten så användes Event Store som en del av lösningen till systemet för att lagra events och data. Event Stores uppgift blev att lagra och presentera data som parametrarna innehöll vilket uppnåddes. I relation till forskningsfrågan, RQ3, så är resultatet för Event Store inte lika positivt för att enbart Event Store kan inte upptäcka anomalier som indikerar på hackingattacker. En begränsning i det utvecklade systemet. Enda sättet för Event Store att upptäcka anomalier vid hackingattacker var att göra det tillsammans med resten av delarna i det utvecklade systemet.

Positiva aspekter med Event Store jämfört med andra databaser är det grafiska gränssnittet, att det är användarvänligt och enkelt att förstå hur events inom Event Store fungerar. Förutom det så sågs ingen fördel med att använda just Event Store i syftet av att upptäcka anomalier som indikerar på attack.

8.2 Jämförelse med relaterat arbete

8.2.1 Studie A

Jämförs det utvecklade systemet med Event Store med systemet från artikeln: Event Stream Database Based Architecture to Detect Network Intrusion [19] så är systemen lika varandra vid funktion såtillvida att de båda använder sig av intrångssystem med event-stream-teknik och databaslagring.

I artikeln så består systemet av en webbplats som förflyttar filer till en databas. Vilket kan liknas vid det utvecklade systemet med Event Store som skriver händelser till en eventdatabas från en hemsida. I artikelns system så använder systemet sig av olika tekniker för t.ex. mönsterrigenkänning och listor med kända hot som jämförs med inkommande events. Som kan jämföras med den utvecklade konsollapplikationen som har framtagna detekteringsregler som identifierar olika hot och potentiella hot mot hemsidan. Jämförs resultatet mellan systemen så möjliggör båda lösningarna att administratören kan definiera egna villkor för att upptäcka attacker, men i systemet från artikeln så arbetar systemet i realtid jämfört med i ej realtid som det utvecklade systemet med Event Store.

8.2.2 Studie B

Det utvecklade systemet med Event Store jämförs med artikeln A study of Methodologies used in Intrusion Detection and Prevention Systems (IDPS) [20] där likheter och skillnader försöker hittas. Artikeln grundar sig på att jämföra de fyra vanligaste metoderna inom detekteringssystem som är avvikelsebaserad, signaturbaserad, tillståndsbaserad och slutligen hybridbaserad.

Likheterna blir tydliga att det utvecklade systemet med Event Store använder sig av ett signaturbaserat system då systemet innehåller en lista på kända hot som genereras baserat på teknisk bevisning, egna antaganden och dummy-data. Ett signaturbaserat system inom artikeln påvisar en stor likhet med systemet inom uppsatsen då systemet enbart känner igen hot inom nuvarande lista över hot och därför har svårt att hitta hot utanför listan. En annan likhet är att systemet är enkelt att konfigurera främst genom följande operationer som att lägga till nya hot och ta bort gamla hot.

En nackdel är att det utvecklade systemet med Event Store är sämre att använda än avvikande detekteringssystem eftersom ett signaturbaserat detekteringssystem är exempelvis svårare att använda och har därmed inte lika bra skydd mot nya attacker.

8.2.3 Studie C

Jämförs utvecklade systemet med Event Store med artikeln Anomaly-based network intrusion detection: Techniques, systems and challenges [21] så kan likheter hittas med att författarna utgår från ett generellt intrångssystem där arkitekturen delas in i fyra block. Blocken kan liknas med det utvecklade systemet är database-boxes som lagrar events och analysis-boxes som analyserar events. I det utvecklade systemet kan detta jämföras med att events lagras i Event Store och eventen utvärderas med hjälp av detekteringsreglerna.

Systemet i artikeln beskrivs som ett avvikelседetekteringssystem vilket skiljer sig mot vad det utvecklade systemet med Event Store använder sig utav som är ett signaturbaserat detekteringssystem. Signaturbaserat detekteringssystem är bra på att upptäcka kända attacker men inte så bra på att upptäcka nya attacker för att det har utvecklats med ett specifikt antal detekteringsregler.

8.2.4 Studie D

Jämförs det utvecklade systemet med Event Store med systemet i artikeln Big Data Analysis System Concept for Detecting Unknown Attacks [22] så har båda system till uppgift att upptäcka och förhindra okända attacker. Systemet i artikeln använder sig av Big Data-analys för att förutspå framtida attacker genom att extrahera data från tidigare okända attacker. Analysen bygger bland annat på fyra tekniker som *förutsägelse*, *klassificering*, *relationsregel* och *atypisk data-mining*.

Likheterna mellan utvecklade systemet med Event Store och systemet i artikeln är tillvägagångssättet. Första likheten är datainsamlingen där data samlas in från databaser, anti-virus, nätverk, system och eventdata från loggar för systemet i artikeln medan data samlas in från ZAP och HTTP Klient från det utvecklade systemet med Event Store. En annan likhet är att data sparas i en dataanordning i artikeln medan data sparas i Event Store för det utvecklade systemet med Event Store. Sista likheten är hur data bearbetas och analyseras, vilket görs via olika tekniker i systemet i artikeln medan detta görs via olika detekteringsregler inom det utvecklade systemet med Event Store.

Olikheterna är teknikerna att känna igen attacker, systemet i artikeln arbetar med att förutspå trender innan systemet blir attackerat medan det utvecklade systemet med Event Store arbetar genom att agera efter att systemet blivit attackerat.

Gemensamt för formen för Machine Learning används inom flertalet av studierna ovan där undersöktes inom uppsatsen ifall det var lämpligt att använda men på grund av tidsbristen inom arbetet så avgränsas arbetet.

9. Slutsats

Syftet med uppsatsen uppfylldes med ett framtaget resultat efter att ha undersökt IT- säkerhet för webbsidor med hjälp av hackingverktyg och att ha undersökt ifall den tekniska lösningen med Event Store möjliggjorde för idén med att hacka dig själv och upptäcka attacker. Svaret per forskningsfråga listas i delkapitel 9.1Frågeställning.

9.1 Frågeställning

9.1.1 RQ1: Hur kan man använda hackingverktyget OWASP Zed Attack Proxy mot egna webbplatser och tjänster för att förbättra det egna säkerhetssystemet?

Hackingverktyget ZAP användes mot det utvecklade systemet med Event Store för att generera attacker som skulle identifieras. Attackerna och andra potentiella attacker upptäcktes med stor sannolikhet genom de framtagna dekeringsreglerna i konsollapplikationen. Sammanfattningsvis så förbättrar ZAP det egna säkerhetssystemet genom att olika anomalier kunde identifieras för attackerna som hade genererats från ZAP.

9.1.2 RQ2: Hur kan man upptäcka att man är under attack eller blivit attackerad?

Det utvecklade systemet med Event Store kan sammanfattningsvis med stor sannolikhet bevisa att attacker har skett baserat på teknisk bevisning och presentation av ZAP och potentiella attacker baserat på den egna kunskapen inom IT-säkerhet och dummy-data från HTTP Klient. Attackerna och potentiella attackerna upptäcktes med de utvecklade detekteringreglerna i konsollapplikationen som reagerade efter anomalier i det utvecklade systemet med Event Store. Resultatet av dekeringsreglerna presenterades i intrångsrapporten från konsollapplikationen. I intrångsrapporten listades alla hot och potentiella hot som det utvecklade systemet med Event Store indikerade.

9.1.3 RQ3: Är Event Store lämpligt att använda för att upptäcka anomalier som indikerar på hackingattacker?

Det upptäcktes att Event Store inte är lämpligt att användas i syftet för att upptäcka anomalier som indikerar på hackingattacker eftersom Event Store inte lyckas upptäcka anomalier på egen hand utan anomalier kunde enbart detekteras tillsammans med resten av det utvecklade systemet. Sammanfattningsvis så konstateras det att Event Store inte är det mest optimala i syftet med detektering av anomalier.

9.2 Förbättringar och framtida utvecklingsmöjligheter

Det som hade underlättat är att i tidigt stadiet inom uppsatsen satt upp en tydlig arbetsprocess och fått en helhetsbild av systemet, för att på så vis begränsa och förhindra att systemet hade kontinuerligt vuxit på bredden och istället arbetat på huvudområden.

Det som hade underlättat processen för att påvisa anomalier för attacker inom Event Store hade varit ifall Event Store innehöll en lista på tidigare kända attacker. Där varje anrop som sker mot webbservern jämförs mot listan för att på ett snabbt sätt kunna påvisa en attack. Med tanke på att Event Store är utvecklat i öppen källkod så kan det antas att listan på kända attacker hade varit omfattande.

En potentiell framtida utvecklingsmöjlighet är hur detekteringsregler bestäms när nya hackingverktyg integreras i systemet. Nuvarande detekteringsregler i dagsläget är anpassade enbart för ZAP och HTTP Klient vilket gör att systemet inte kommer indikera på attacker ifall nya hackingverktyg används eftersom det då saknas regler.

En annan utvecklingsmöjlighet är att systemet i dagsläget inte indikerar attacker i realtid utan detta sker i efterhand, allteftersom detekteringsreglerna skrivs. Skulle systemet kunna utökas med en databas som innehåller kända hot så underlättar detta för administratören av systemet i uppsatsen och reducerar tiden för administratören att analysera inkommande trafik emot systemet. Även fel i form av mänskliga faktorn kan reduceras markant.

Avslutningsvis är själva representationen av resultatet en utvecklingsmöjlighet. I dagsläget görs den visuella representationen enkelt i form av ett docx-format där detekteringsreglerna tas upp och antalet förfrågningar per regel. Vilket diskuterades med uppdragsgivaren i början av uppsatsen om att ta fram en komplett dashboard med olika grafer och tabeller om vilka attacker som skett, hur ofta, vilket tidspunkt m.m (se intrångsrapporten i Appendix D). Efter diskussion med uppdragsgivaren så var representation av en dashboard en för stor uppgift och det hade varit ett eget examensarbete.

Referenser

- [1] Stuttard, D., & Pinto, M. (2011). *The web application hackers handbook: Discovering and exploiting security flaws* (2nd ed.) [2]. Retrieved June 2, 2019.
- [2] Fallon, N. (2014, January 06). 10 Ways to Improve Your IT Security in 2014 | IT Security. Retrieved June 3, 2019, from <http://www.businessnewsdaily.com/5717-improve-it-security.html>
- [3] Troy Hunt. (2013, May 15). Hack yourself first – how to go on the offence before online attackers do. Retrieved June 3, 2019, from <https://www.troyhunt.com/hack-yourself-first-how-to-go-on>
- [4] Our Most Advanced Penetration Testing Distribution, Ever. (n.d.). Retrieved June 3, 2019, from <http://www.kali.org/>
- [5] Penetration Testing Software, Pen Testing Security. (n.d.). Retrieved June 3, 2019, from <http://www.metasploit.com/>
- [6] OWASP Zed Attack Proxy Project. (n.d.). Retrieved June 7, 2019, from https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
- [7] What is a White Hat Hacker? - Definition from Techopedia. (n.d.). Retrieved June 3, 2019, from <https://www.techopedia.com/definition/10349/white-hat-hacker>
- [8] What is a Black Hat Hacker? - Definition from Techopedia. (n.d.). Retrieved June 3, 2019, from <https://www.techopedia.com/definition/26342/black-hat-hacker>
- [9] Alsunbul, S., Le, P., Tan, J., & Srinivasan, B. (2016). A network defense system for detecting and preventing potential hacking attempts. *2016 International Conference on Information Networking (ICOIN)*, 449-454. doi:10.1109/icoin.2016.7427157
- [10] Event Sourcing Basics. (n.d.). Retrieved June 3, 2019, from <https://eventstore.org/docs/event-sourcing-basics/>
- [11] NEventStore. (n.d.). NEventStore/NEventStore. Retrieved June 3, 2019, from [https://github.com/NEventStore/NEventStore/wiki/Architectural Overview](https://github.com/NEventStore/NEventStore/wiki/Architectural%20Overview)
- [12] Webserver. (n.d.). Retrieved May 5, 2019, from <http://www.ipeer.se/webbserver.php>
- [13] HTTP Request Methods. (n.d.). Retrieved May 5, 2019, from http://www.w3schools.com/tags/ref_httpmethods.asp
- [14] Event Store. (n.d.). Retrieved June 3, 2019, from <http://geteventstore.com/>
- [15] Complex Event Processing. (n.d.). Retrieved May 5, 2019, from <https://databricks.com/glossary/complex-event-processing>
- [16] Menegaz, G. (2012, October 01). What is NoSQL, and why do you need it? Retrieved June 3, 2019, from <http://www.zdnet.com/what-is-nosql-and-why-do-you-need-it-7000004989/>

- [17] SQLCourse. (2000, August 20). What is SQL? Retrieved May 5, 2019, from <http://www.sqlcourse.com/intro.html>
- [18] Penetrationstest | IDG:s ordlista. (n.d.). Retrieved May 19, 2019, from <https://it-ord.idg.se/ord/penetrationstest/>
- [19] Kumaran, V. (2013). Event stream database based architecture to detect network intrusion. *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems - DEBS 13*, 241-247. doi:10.1145/2488222.2488276
- [20] Mudzingwa, D., & Agrawal, R. (2012). A study of methodologies used in intrusion detection and prevention systems (IDPS). *2012 Proceedings of IEEE Southeastcon*, 1-6. doi:10.1109/secon.2012.6197080
- [21] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2), 18-28. doi:10.1016/j.cose.2008.08.003
- [22] Ahn, S., Kim, N., & Chung, T. (2014). Big data analysis system concept for detecting unknown attacks. *16th International Conference on Advanced Communication Technology*, 269-272. doi:10.1109/icact.2014.6778962
- [23] Controlled experiments. (n.d.). Retrieved June 7, 2019, from <https://www.khanacademy.org/science/high-school-biology/hs-biology-foundations/hs-biology-and-the-scientific-method/a/experiments-and-observations>
- [24] What is an End-to-End Test? - Definition from Techopedia. (n.d.). Retrieved June 3, 2019, from <https://www.techopedia.com/definition/7035/end-to-end-test>
- [25] OWIN. (n.d.). Retrieved June 3, 2019, from <http://owin.org/>
- [26] Tutorialspoint.com. (n.d.). MVC Framework Tutorial. Retrieved June 3, 2019, from https://www.tutorialspoint.com/mvc_framework/
- [27] BillWagner. (n.d.). Asynchronous programming in C#. Retrieved June 3, 2019, from <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/>
- [28] Web Parameter Tampering. (n.d.). Retrieved June 3, 2019, from https://www.owasp.org/index.php/Web_Parameter_Tampering
- [29] Format string attack. (n.d.). Retrieved June 3, 2019, from https://www.owasp.org/index.php/Format_string_attack
- [30] Vad är streaming? (n.d.). Retrieved June 3, 2019, from <https://www.prv.se/sv/upphovsratt/streama-lagligt/vad-ar-streaming/>
- [31] Streaming - på Sveriges största ordbok. (n.d.). Retrieved June 3, 2019, from <https://www.synonymer.se/sv-syn/streaming>
- [32] SQL Injection. (n.d.). Retrieved April 28, 2019, from https://www.owasp.org/index.php/SQL_Injection

- [33] Testing for SQL Injection (OTG-INPVAL-005). (n.d.). Retrieved April 28, 2019, from [https://www.owasp.org/index.php/Testing_for_SQL_Injection_\(OTG-INPVAL-005\)](https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005))
- [34] Testing for Format String. (n.d.). Retrieved April 28, 2019, from https://www.owasp.org/index.php/Testing_for_Format_String
- [35] Owasp. (n.d.). OWASP/CheatSheetSeries. Retrieved April 10, 2019, from https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md
- [36] Testing for CSRF (OTG-SESS-005). (n.d.). Retrieved April 10, 2019, from [https://www.owasp.org/index.php/Testing_for_CSRF_\(OTG-SESS-005\)](https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005))
- [37] What Is SSL (Secure Sockets Layer)? (n.d.). Retrieved April 28, 2019, from <https://www.digicert.com/ssl/>
- [38] Cross-site Scripting (XSS). (n.d.). Retrieved June 3, 2019, from [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [39] Forced browsing. (n.d.). Retrieved June 3, 2019, from https://www.owasp.org/index.php/Forced_browsing
- [40] Choudhary, V. (2018, March 29). Difference between C and Ansi C. Retrieved May 16, 2019, from <https://developerinsider.co/difference-between-c-and-ansi-c/>
- [41] Vad är ascii - IT-ord från Computer Sweden. (n.d.). Retrieved May 16, 2019, from <https://it-ord.idg.se/ord/ascii/>
- [42] What is complex event processing (CEP)? - Definition from WhatIs.com. (n.d.). Retrieved June 2, 2019, from <https://searchmicroservices.techtarget.com/definition/complex-event-processing-CEP>
- [43] What is RDBMS (relational database management system)? - Definition from WhatIs.com. (n.d.). Retrieved June 2, 2019, from <https://searchdatamanagement.techtarget.com/definition/RDBMS-relational-database-management-system>
- [44] What is a localhost? (n.d.). Retrieved June 3, 2019, from <https://whatismyipaddress.com/localhost>
- [45] What is Representational State Transfer (REST)? - Definition from Techopedia. (n.d.). Retrieved June 7, 2019, from <https://www.techopedia.com/definition/1312/representational-state-transfer-rest>
- [46] Overview. (n.d.). Retrieved February 20, 2019, from <https://eventstore.org/docs/http-api/index.html>

Appendix A

Versionen av Event Store som användes i uppsatsen var den exekverbara versionen med versionsnumret 5.0.0.

Den exekverbara versionen av Event Store startas genom kommandotolken i Windows-operativsystem. Kommandot: `EventStore.SingleNode.exe --db ./db --log ./logs`, anges i kommandotolken efter att ha navigerat i kommandotolken till mappen med alla tillhörande filer för Event Store. När Event Store startas så körs Event Store som en server. Standard så körs Event Store på port 2113. Kommandot: `EventStore.SingleNode.exe --db ./db --log ./logs`, startar den exekverbara filen `EventStore.SingleNode.exe`. Texten `--db ./db --log ./logs` innebär att databasen placeras i mappen `db` och att loggningsfilerna sparas i mappen `logs`. Om webbadressen `127.0.0.1` (localhost) anges i en webbläsare med port 2113 så hämtas det grafiska gränssnittet som tillhör Event Store: `http://localhost:2113` eller `http://127.0.0.1:2113`. Se Figur 9.



Figur 9, Event Store GUI

Figuren visar hur Event Stores webbgränssnitt ser ut i menyn längst upp till höger.

För att kommunicera med Event Store så används ett REST-baserat API. REST står för Representational State Transfer och är ett distribuerat ramverk som använder olika tekniker och webbprotokoll. Den innehåller klient- och serverinteraktioner som byggs kring överföringen av resurser. Ett system som innehåller följer REST-principer brukar benämnas som RESTful [45].

Det REST-baserade API:t som används är AtomPub-protokollet. Varför AtomPub-protokollet är en del av kommunikationen för Event Store är för att alla utomstående system eller miljöer ska kunna integreras på ett enkelt sätt med Event Store. AtomPub-protokollet ger en fördel att Event Store kan användas i miljöer med komponenter som har olika drifttider. Exempel på detta är ifall Event Store skulle användas tillsammans med SmallTalk, C# och Haskell accesstreams med events. En nackdel med AtomPub-protokollet är att responstiden är stor jämfört med andra protokoll [46].

Event Store stödjer ett flertal format för de inkommande HTTP-förfrågningarna t.ex. `application/xml` och `application/json`. I HTTP-förfrågningar mot Event Store så anges det i URL:en mot Event Store vilket format som ska användas [46].

Appendix B

Inuvarande appendix så beskrivs de olika hackingattackerna som inte har blivit beskrivna i teknologiasvnittet.

SQL-injection

Attacken (SQL Injection) görs genom att en administratör skickar med en SQL-fråga mot en applikation vid inmatning av data [32].

Cross-site Scripting

När skadliga skript injiceras mot en hemsida så kallas attacken för Cross-site Scripting. Attackerna inträffar när en administratör av en specifik hemsida får skadliga skript skickade mot hemsidan av en hackare. Cross-site scripting förkortas också XSS [38].

Forced browsing

Forced Browsing är en typ av attack som baseras på förutspådda filer eller direktivnamn i URL:en för en hemsida som t.ex. `www.site-example.com/users/calendar.php/user1/20070715`. Hackare kan gissa sig fram till en annan slutanvändares inloggning genom användarinformation. I exemplet så är `user1` och `20070715` olika exempel på användarinformation [39].

Appendix C

I appendix så finns kod hur HTTP Klient fungerar för att skicka HTTP POST- förfrågningar mot hemsidan.

```
static async Task RunAsync()
{
    client.BaseAddress = new Uri("http://localhost:16911/");
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

    try
    {
        // Create a new product
        Product product = new Product { Name = "Gizmo", Price = 100, Category = "Widgets" };
        int x = 0;
        while(x <= 10)
        {
            var result = client.PostAsJsonAsync<string>("TestHAHAHA", "Hello World").Result;
            x++;
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }

    Console.ReadLine();
}
}
```

Figur 10, HTTP Klient Program

Appendix D

Intrångsrapport för hemsidan

De olika hoten som har hittats är:

Content-length paramtern är större än 0, har hittats: 19 gånger från HTTP Klient

Pragma parametern finns med i vissa events. Den har hittats: 2671 gånger från ZAP

Värdet ZAP för ett events URL indikerar på en attack. Det har hittats: 42 gånger från ZAP

Värdet owasp för ett events URL indikerar på en attack, har hittats: 36 gånger från ZAP

Attacken CSRV – Ordet search har hittats: 24 gånger från ZAP

Attacken HTTP Only – Ordet https har hittats: 16 gånger från ZAP

Attacken Format String Error – Ordet print har hittats: 20 gånger från ZAP

Attacken SQL Injection – Ordet passwd har hittats: 36 gånger från ZAP

Figur 11, Intrångsrapporten