CHALMERS

Sakernas internet: Cyberhot och möjliga motåtgärder

Examensarbete inom Data- och Informationsteknik

Dennis Ek

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2018

EXAMENSARBETE

Sakernas internet: Cyberhot och möjliga motåtgärder

Dennis Ek

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2018

Sakernas internet: Cyberhot och möjliga motåtgärder
Dennis Ek

Examinator: Jonas Duregård

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Institutionen för Data- och Informationsteknik
Göteborg 2018

Sakernas internet: Cyberhot och möjliga motåtgärder

DENNIS EK
Institutionen för Data- och Informationsteknik, Chalmers tekniska högskola

Kandidatarbete

Sammanfattning

Användningen av sakernas internet har på senare tid ökat i rask takt. Genom att koppla upp produkter mot internet ges en organisation tillgång till ny funktionalitet som inte varit möjlig tidigare, men genom att koppla upp enheter mot internet blottläggs samtidigt dessa enheter för nya illasinnade aktörer som försöker ta kontroll över enheterna. Ett företag som använder sig av sakernas internet är Picadeli AB, vars företagsidé är att sälja sallad i lösvikt till konsumenter. Detta uppnås genom att Picadeli placerar salladsbarer hos butiker, varav de senaste generationerna av salladsbarer är anslutna till internet. Denna internetanslutning gör därmed Picadeli till en användare av sakernas internet (eng. Internet of Things), och följaktligen även en måltavla för illasinnade angripare. Genom användning av Microsoft Threat Modeling Tool 2016, samt kompletterande informationsinhämtning från de senaste årens forskningslitteratur har en lista över hot mot enheter i ett sakernas internet tagits fram, och utifrån denna lista har hotens relevans för Picadeli utvärderats. Resultat och lösningar på problemen som påvisats i denna utvärdering presenteras här, följt av generella riktlinjer — generaliserade utifrån listan över hot — som samtliga organisationer bör följa för att undvika säkerhetsbrister i sin infrastruktur för sakernas internet. Därmed är resultatet till nytta även för andra företag än Picadeli.

Nyckelord: Internet of Things, IoT, sakernas internet, säkerhet, datasäkerhet

Innehåll

1. *Inledning*

1.1 Bakgrund

Sakernas internet (eng. Internet of Things) är ett namn på de sammankopplade system som består av enheter som med hjälp av givare eller ställdon kan samla in information om omgivningen respektive påverka densamma [1]. Sakernas internets mångsidighet har gjort att det anammats av många olika organisationer, med vitt spridda användningsområden [2].

Det svenska livsmedelsföretaget Picadeli AB är ett företag vars affärsidé går ut på att via matbutiker samt egna restauranger sälja sallad på lösvikt till konsumenter. Genom att placera salladsbarer i redan existerande matbutiker kommer de i kontakt med sin målgrupp, alltmedan matbutikerna kan öka sin omsättning och potentiellt attrahera nya kunder.

De senaste åren har Picadeli placerat den internetuppkopplade modellen Arctic i butiker, vilken sedermera ersatts av den förbättrade modellen Nordic. Genom att ständigt vara uppkopplade mot internet — och därmed utgöra ett sakernas internet — kan matkvaliteten ytterligare garanteras, bland annat genom realtidsövervakning av driftdata och förfluten tid sedan maten placerats i salladsbaren.

1.2 Syfte

Allt eftersom fler och fler har börjat använda sakernas internet har det samtidigt blivit ett lovligt byte för illasinnade aktörer [3]. Genom att förhindra intrång i systemen kan en organisation undvika att känsliga uppgifter läcker ut, eller att hela system slås ut. Skulle olyckan väl vara framme kan det få såväl politiska som sociala och ekonomiska följder.

För Picadelis räkning är det av högsta intresse att deras infrastruktur står emot cyberangrepp från angripare. Då Picadelis övervakning av salladsbarerna förlitar sig på att salladsbarerna rapporterar data med jämna mellanrum och att datan som rapporteras inte är manipulerad på något sätt, är det av stort intresse för Picadeli att

salladsbarerna står emot cyberangrepp från illasinnade aktörer. Vidare ska datan som skickas, utöver att vara oförvanskad, endast vara tillgänglig för anställda på Picadeli.

I en tid då cyberangrepp och -spionage blir allt vanligare är det viktigt för organisationer att ha en tydligt definierad strategi för att försvara sig mot angripare. Slutsatserna från denna studie kommer därför inte vara relevanta endast för Picadeli, utan även för myndigheter såväl som andra företag och organisationer.

1.3 Mål

Projektet kommer att presentera riktlinjer för en säker infrastruktur för sakernas internet utifrån en fallstudie genomförd hos Picadeli. I dessa riktlinjer kommer även en handlingsplan att ingå, vars syfte är att underlätta återhämtning för en organisation om en illasinnad aktör trots allt skulle åsamka skada på en eller flera delar av organisationens system.

Vidare kommer projektet även att undersöka huruvida Microsoft Threat Modeling Tool 2016, som är ett verktyg framtaget för att studera säkerheten hos mjukvaruprojekt, även lämpar sig för att utvärdera säkerheten inom ett sakernas internet.

Om en eller flera säkerhetsbrister skulle upptäckas hos Picadeli under projektets gång, kommer dessa att åtgärdas i samverkan med Picadeli. Lösningarna på dessa säkerhetsbrister kommer att presenteras på ett sådant sätt i rapporten, så att även andra organisationer kan dra nytta av lärdomarna och implementera skydd mot de typer av angrepp som beskrivs.

1.4 Avgränsningar

Eventuella säkerhetsbrister hos Picadeli vars lösningars tidsåtgång bedöms ligga utanför projektets tidsrymd kommer inte att hanteras — varken i dokumentation eller i form av lösningsförslag. Vidare kommer Picadelis miljö vara den enda fallstudie som genomförs, och av den kommer endast delarna som berör sakernas internet att undersökas.

2. *Teori*

Nedan kommer de kunskapsområden och begrepp, som läsaren behöver för att förstå resten av rapporten, att presenteras. Först ges en kort introduktion till sakernas internet, följt av ett antal vanliga tekniker som används inom detsamma. Därefter presenteras säkerhetsmodellen STRIDE, samt verktyget Microsoft Threat Modeling Tool 2016 som förlitar sig på denna modell.

2.1 Sakernas internet

Ingenjörsorganisationen IEEE definierar sakernas internet som ett nätverk av sammankopplade enheter med särskilda egenskaper [1]. Rapporten kommer använda denna definition, och kommer därför att här nedan presentera några av de viktigaste av dessa egenskaper som IEEE definierat.

För att de sammankopplade enheterna ska utgöra ett sakernas internet är det nödvändigt att de är unikt identifierbara inom ett givet system. Dessa enheter är dessutom kopplade till internet, och tack vare denna egenskap är det möjligt att läsa av och påverka tillståndet hos enheten från fjärran.

För att kunna ta in data från omgivningen, samt påverka densamma, är dessa enheter utrustade med givare respektive ställdon. Dessa enheter kan vidare ha en möjlighet att anpassa funktionaliteten som en reaktion på samspel med en användare, eller som en reaktion på yttre stimuli genom en givare.

Idag används sakernas internet inom många fält. Bland annat används det till att automatiskt mäta elförbrukning i smarta elnät; i kommunikation mellan fordon för att kommunicera om hur läget ser ut längre bort än vad som syns med kameror och dylikt; samt inom hälsoappplikationer såsom smarta armband, och medicinska implantat [3].

2.2 SSH

Secure Shell (SSH) är ett protokoll för säker kommunikation över osäkra medier, såsom internet. Protokollet består av tre delprotokoll, varav det första (SSH Transport Layer Protocol) ansvarar för att autentisera servern, erbjuda konfidentialitet samt integritet. Det andra protokollet (SSH Authentication Protocol) används när en klient ska autentisera sig mot en server. Det tredje och sista protokollet (SSH Connection Protocol) har som uppgift att exponera logiska kanaler mellan servern och klienten [4].

SSH togs fram då dess föregångare (bland andra Telnet och rlogin) inte erbjöd användaren de fördelar som kryptering erbjuder. Detta gjorde det möjligt för en angripare att anfalla ett system relativt obehindrat, vilket kunde förhindras genom användandet av SSH.

En klient kan autentisera sig mot en server med hjälp av ett antal metoder. En av dessa metoder sker med hjälp av ett lösenord. Här förser klienten servern med ett användarnamn och ett lösenord, varpå servern bedömer uppgifterna och svarar klienten med information om inloggningen lyckats eller ej. En annan metod för klienten att autentisera sig är med hjälp av assymetrisk kryptering, vilket innefattar ett nyckelpar där den ena är offentlig och den andra är privat. Här skapar klienten en signatur med hjälp av sin privata nyckel, varpå servern validerar signaturen med hjälp av klientens publika nyckel, samt kontrollerar att den publika nyckeln är en tillåten autentiseringsmetod för användaren [5].

2.3 VPN och OpenVPN

Genom att använda sig av ett virtuellt privat nätverk (VPN) kan enheter på flera olika platser kommunicera med varandra på samma sätt som om de befunnit sig på samma lokala nätverk. Vidare kan dessa tjänster erbjuda möjligheten till säkra kommunikationskanaler över annars osäkra medium. Tack vare detta lämpar sig tekniken väl för att användas i samband med sakernas internet.

En mjukvara för att sätta upp ett VPN är OpenVPN. OpenVPN erbjuder möjligheten att sätta upp ett VPN antingen på nivå 2 eller nivå 3 i OSI-modellen. OSI-modellen är en modell som tagits fram för att dela upp nätverkskommunikation i ett antal distinkta lager. På nivå 2 återfinns datalänkslagret, som har till uppgift att transportera data över en länk — exempelvis genom en nätverkskabel eller från en trådlös sändare till en mottagare. En nivå upp, på nivå 3, finns nätverkslagret. I nätverkslagret ryms protokoll för att dirigera nätverksdata från en enhet till en annan, över ett flertal datalänkar. Genom att arbeta på nivå 2 erbjuder OpenVPN samma funktionalitet som en nätverksswitch, vilket medför stöd även för speciella nätverkstjänster som använder sig av exempelvis specialkonstruerade ethernetra-

mar. I de fall då det inte finns ett behov av dessa tjänster konfigureras med fördel OpenVPN till att arbeta på nivå 3 i OSI-modellen. Genom att inte kapsla in trafiken i en ethernetheader minskar mängden data som skickas över VPN:et, och medför därmed bättre prestanda till en kostnad av att stödja färre nätverkstjänster. Trafik mellan olika OpenVPN-noder sker över en kontrollkanal, och en datakanal — båda över samma TCP-port [6].

Beroende på om VPN:et arbetar på nivå 2 eller 3 i OSI-modellen kommer OpenVPN att skapa en virtuell TAP- respektive TUN-nätverksenhet. Benämningen TAP kommer från network tap, för att illustrera dess funktion — att koppla in sig på ett dataflöde och börja lyssna på trafiken (eng. tap into). Här skapas en nätverksbrygga som bryggar samman det virtuella nätverket med enhetens fysiska. TUN å andra sidan kommer från att en nätverkstunnel skapas. Här dirigeras trafik till specificerade nätverk över denna nätverkstunnel, med hjälp av tekniker så som IP. I likhet med ett fysiskt nätverkskort, kan dessa virtuella nätverksenheter filtreras med hjälp av en brandvägg [7].

För att skydda trafik mellan två OpenVPN-noder använder sig OpenVPN av TLS på kontrollkanalen för att autentisera enheter samt utbyta nycklar för kryptering av datakanalen. TLS är ett protokoll för krypterad nätverkskommunikaion, och erbjuder som sådant konfidentialitet och dataintegritet för nätverkstrafiken. Genom att använda sig av TLS kan OpenVPN dessutom portas mellan olika plattformar, då det finns välstuderade implementationer av TLS på de flesta plattformar [8].

2.4 STRIDE

STRIDE är en modell framtagen av Microsoft för att kategorisera hot mot IT-resurser. Namnet är en initialförkortning bestående av begreppen Spoofing (sv. förfalskning), Tampering (sv. manipulering), Repudiation (sv. förnekande), Information Disclosure (sv. utlämnande av information), Denial of Service (sv. tillgänglighetsattack) och Elevation of Privilege (sv. behörighetshöjning), vilka var för sig eller i kombination kan utgöra ett hot [9]. Här nedan kommer dessa begrepp att beskrivas, i enlighet med Microsofts definitioner.

Spoofing handlar om att utge sig för att vara någon annan, exempelvis med hjälp av inloggningsuppgifter som en angripare kommit över. Det kan också röra sig om ett angrepp på en autentiseringsmekanism med svagt eller inget skydd, för att på så sätt tillskansa sig de rättigheter som en särskild användare har.

Med *tampering* avses ändringar på data, på ett sådant sätt som inte är tillåten enligt rådande policy. Ett exempel skulle kunna vara ett virus som sprider sig själv till andra filer när en användare startar ett program som infekterats av viruset.

Repudiation åsyftar anonymitet i handlingar. Till exempel kan det röra sig om en angripare som sopar igen spåren efter sig, för att en eventuell utredning inte ska

kunna leda utredaren till angriparen.

Med hjälp av *Information Disclosure* kan en angripare komma över data som inte bör vara tillgänglig för andra än behöriga användare. Här skulle ett angrepp kunna bestå av en angripare som utnyttjar en felkonfigurerad webbserver, som tillåter åtkomst till andra filer än de filer som hör till en viss hemsida.

En hemsida som utsätts för en överbelastningsattack är utsatt för en *Denial of Service*-attack. Ett *Denial of Service*-hot är ett hot som riskerar att neka en legitim användare tillgång till den data som användaren är auktoriserad att komma åt.

Slutligen är *Elevation of Privilege* ett sätt för en angripare att utöka sina rättigheter på systemet. Genom att utnyttja sårbarheter i ett operativsystem skulle exempelvis en användare kunna tillskansa sig fler behörigheter än vad en administratör hade avsett.

2.5 Microsoft Threat Modeling Tool 2016

Threat Modeling Tool 2016 är ett verktyg från Microsoft som tagits fram för att upptäcka sårbarheter i mjukvara tidigt i utvecklingsprocessen [10]. Verktyget låter en användare grafiskt modellera ett system med hjälp av ett antal fördefinierade grundobjekt, samt att skapa egna objekt med användardefinierade egenskaper. När modellen sedan bedöms stämma överens med systemet som modelleras, genererar verktyget automatiskt en lista över potentiella sårbarheter i systemet, där varje hot klassificeras i enlighet med säkerhetsmodellen STRIDE.

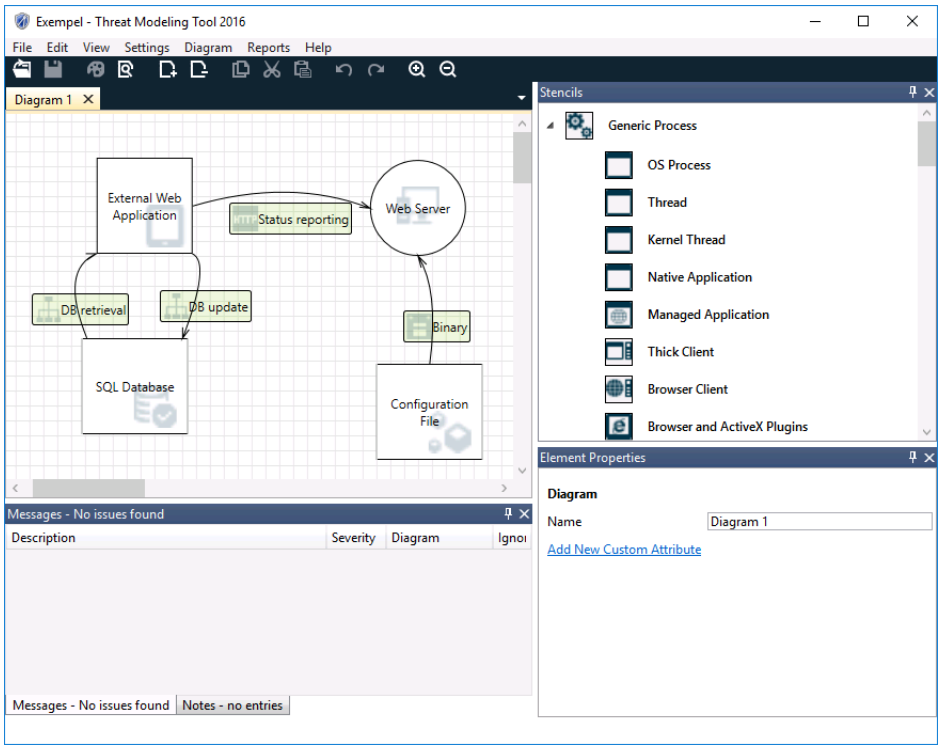
När användaren skapat ett projekt presenterar verktyget designvyn för användaren (se figur 2.1). I denna vy återfinns ett fönster för diagrammet samt tre andra fönster. Från fönstret *Stencils* kan användaren placera ut objekt i diagrammet genom att dra dem från *Stencils* och släppa dem i diagrammet. Under det fönstret finns *Element Properties* (se nedan för en beskrivning av detta fönster), och till vänster om det finns fönstret *Messages*. I *Messages* presenteras de varningar och felmeddelanden som verktyget eventuellt har upptäckt.

Genom att klicka på ett objekt i diagrammet ges användaren möjlighet att ändra ett objekts egenskaper. I figur 2.2 har användaren markerat objektet *Web Server*. I fönstret *Element Properties* syns då de egenskaper som användaren kan ändra hos objektet. I detta fall kan till exempel användaren ange att data till och från webbservern filtreras med hjälp av inställningarna för *Sanitizes Input* och *Sanitizes Output*.

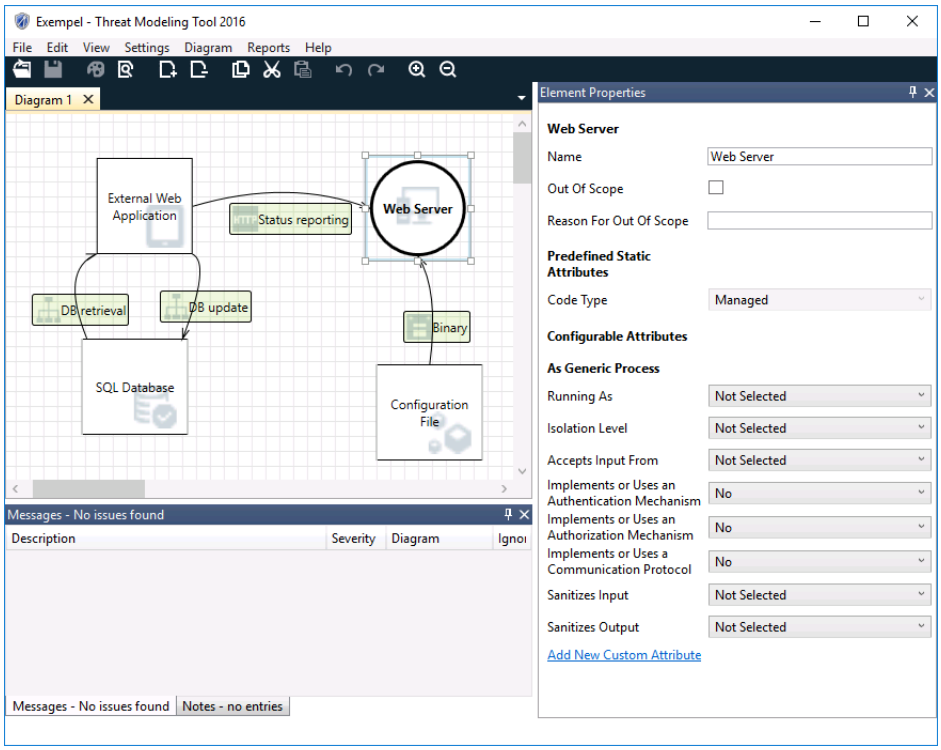
När allting sedan är färdigt, kan användaren växla till analysvyn i verktygsraden. Då presenteras två nya fönster för användaren — som går att skåda i figur 2.3. I det översta fönstret, *Threat List*, visas en lista med potentiella hot. I det undre fönstret, *Threat Properties*, får användaren en utförlig beskrivning av det ur listan

markerade problemet, samt möjligheter att hantera det. Här finns möjligheten att markera potentialla hot som *ej påbörjat*, *inte applicerbart*, *i behov av undersökning*, samt *åtgärdat*.

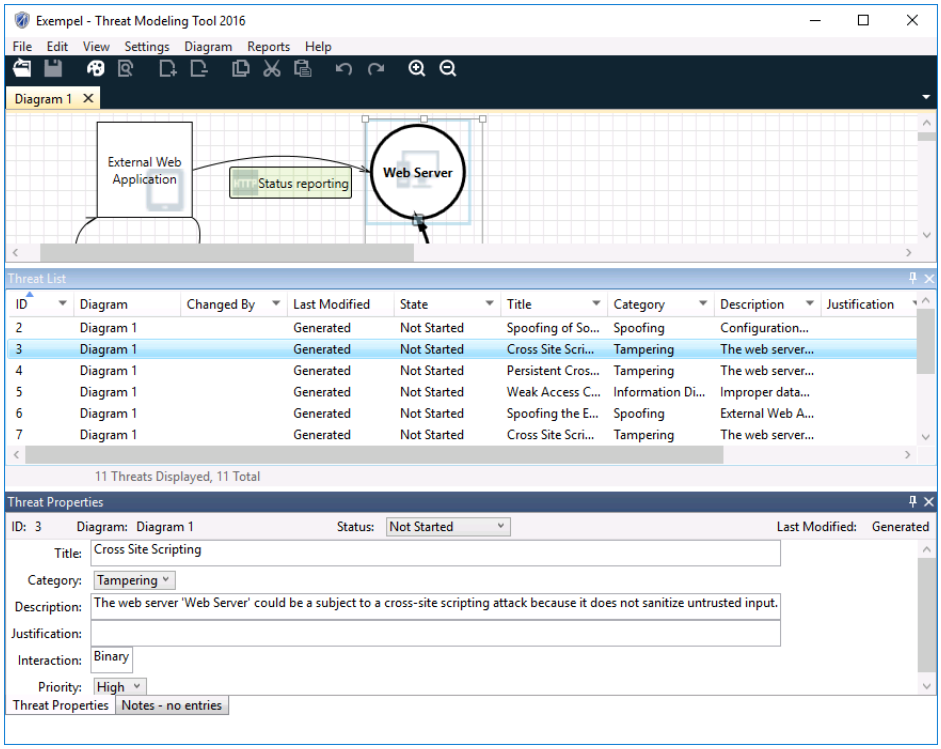
Inbyggt i verktyget är även funktionalitet för att generera en lättläst rapport. Detta finns tillgängligt i menyn *Reports*, och rapporterna därifrån kan användas som underlag för planering av vidare utveckling.



Figur 2.1: Designvyn i Threat Modeling Tool 2016. Uppe till vänster syns diagrammet, det vill säga det modellerade systemet. Nere till vänster syns *Messages*-fönstret där eventuella varningar och felmeddelande presenteras. Uppe till höger finns en lista över grundobjekt som går att placera i diagrammet, och nere till höger syns ett fönster för att redigera ett markerat objekts egenskaper.



Figur 2.2: Fönstret *Element Properties* i Threat Modeling Tool 2016. Här har ett objektet *Web Server* markerats, och till höger i bild finns fönstret *Element Properties*, där de inställningar som går att göra på objektet visas.



Figur 2.3: Analysvyn i Threat Modeling Tool 2016. Överst syns diagrammet, och under det visas en lista över de potentiella hot som upptäckts av programmet. Nederst visas fönstret *Threat Properties* där en användare ändrar status och kommenterar på hoten.

3. *Metod*

De delar av Picadelis infrastruktur som berör sakernas internet kommer att modelleras i Microsoft Threat Modeling Tool 2016. De inbyggda standardkomponenterna kommer att användas för att så verklighetstroget som möjligt modellera Picadeli, och utifrån denna modell kommer en fullständig rapport med potentiella sårbarheter att genereras. Vidare kommer även verktygets lämplighet för att utvärdera säkerheten inom sakernas internet att utvärderas, då verktyget i första hand är ett verktyg för att modellera mjukvara, och inte en hel infrastruktur. Utifrån rapporten med möjliga sårbarheter, som verktyget genererar efter avslutad modellering, kommer rapporten att granskas manuellt för att sålla bort falska alarm. Om någon av posterna i rapporten skulle visa sig utgöra en säkethetsrisk kommer ett lösningsförslag att tas fram åt Picadeli.

Vidare kommer resultatet från Threat Modeling Tool att kompletteras med hot som beskrivits i samtida litteratur. Även här kommer hoten att analyseras, och om någon av dessa visar sig utgöra en säkerhetsrisk för Picadeli kommer även här ett lösningsförslag att presenteras.

3.1 Microsoft Threat Modeling Tool 2016

Med hjälp av Microsoft Threat Modeling Tool 2016 modellerades Picadelis infrastruktur enligt figur 3.1 på sida 13. Modelleringen kan delas in i tre stycken huvuddelar: salladsbar, back-end och användare.

Till vänster i figur 3.2 återfinns en modell av en salladsbar. I varje salladsbar finns det en huvudmodul — AU — som är ansvarig för kommunikation med omvärlden. På denna AU finns det även en databas som lagrar salladsbarens nuvarande tillstånd. Vidare finns det ett antal kylskåpsmoduler — FCU:er — beroende på salladsbarens storlek, som kommunicerar med AU:n. Denna kommunikation består bland annat av temperaturrapportering och information om var olika produkter befinner sig. Till sist finns det även en surfplatta som hämtar information från AU:n som sedan presenteras grafiskt på en skärm i salladsbaren.

Till höger i modellen återfinns Picadelis back-end som varje salladsbar kommunice-

rar med, se figur 3.3. Kommunikationen mellan webbservern och databasen, samt databasen i sig är ej exponerade mot internet, och därför är den enda del av denna back-end som berör sakernas internet gränssnittet som webbservern tillhandahåller. Av denna anledning har därför databasen samt kommunikationen mellan den och webbservern markerats som ointressant för modelleringen, med hjälp av inställningen "Out of scope" i verktyget.

Nederst i modellen syns till sist en modell av en användare. Med hjälp av SSH har en användare tillgång till en salladsbars moduler. Denna anslutning sker över ett VPN som inte modellerats, då det bedömdes att detta VPN enbart bidrog till mer komplexitet i modellen utan att bidra till en mer detaljerad analys. Anledningen till att ett modellerat VPN inte skulle bidra med mer detaljer är för att SSH redan erbjuder samma säkerhet som ett VPN gör i detta fall.

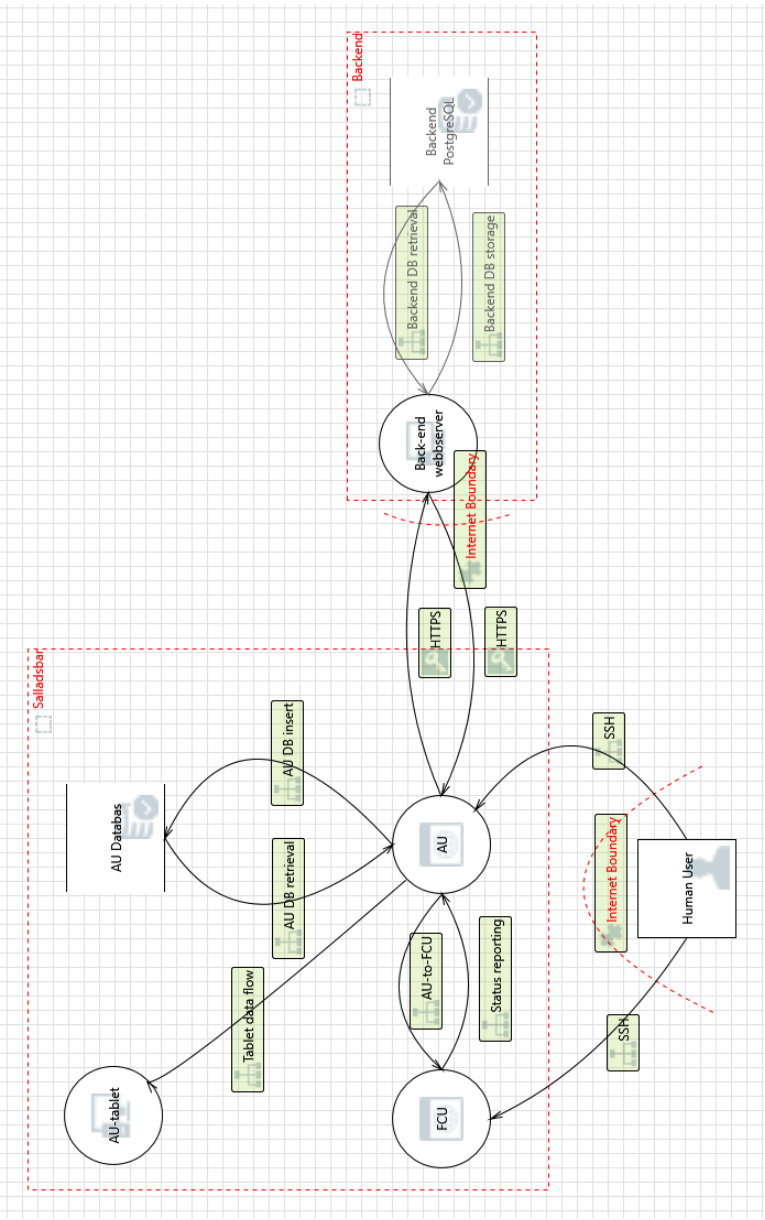
Kommunikation mellan varje AU och Picadelis back-end äger rum över en HTTPS-anslutning. Denna anslutning är modellerad med hjälp av verktygets fördefinierade objekt för HTTPS-kommunikation. Resterande dataflöden är modellerade med hjälp av objekt av typen *Generic Data Flow* vars egenskaper har ställts in för att efterlikna verkligheten.

3.2 Datainsamling ur litteratur

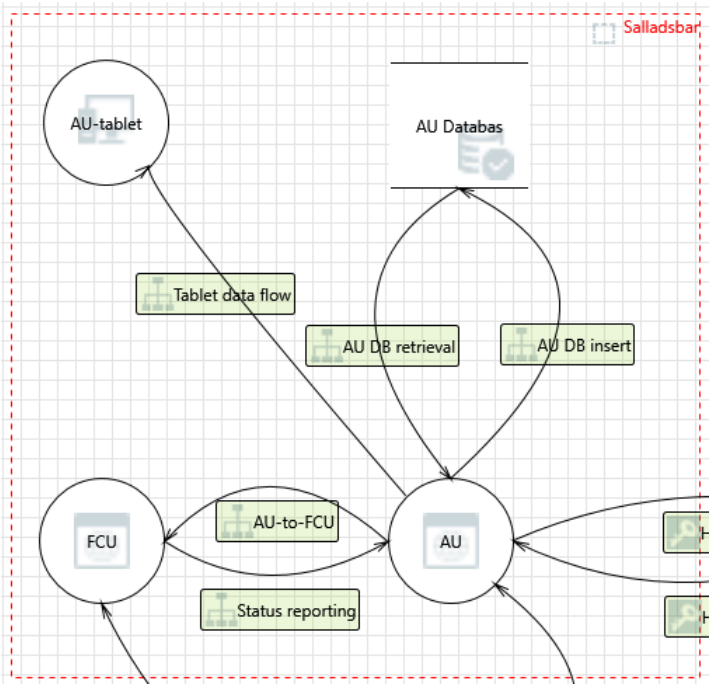
Sökningar kommer att genomföras i de databaser som tillhandahålls av Chalmers e-bibliotek. Bland annat kommer sökningar att göras i databaserna IEEE Xplore, som innehåller material från organisationerna IEEE och IETF.

Med hjälp av nyckelord som bedöms relevanta inom ramen för säkerhet inom sakernas internet kommer material på svenska och engelska undersökas.

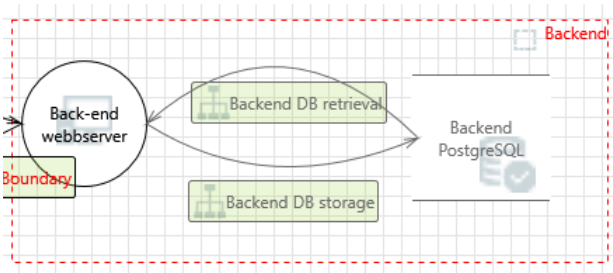
Då även en fallstudie kommer att utföras kommer antalet resurser hämtade med hjälp av denna metod begränsas till fem stycken.



Figur 3.1: Modell över Picadelis sakernas internet. Till vänster är en salladsbar modellerad, med huvudmodulen AU, kylskåpsmodulen FCU, en databas, samt en surfplatta. Till höger återfinns Picadelis webbserver dit salladsbarerna kommunicerar, och nederst i bild finns en modell av en användare som loggar in på salladsbaren med hjälp av SSH.



Figur 3.2: Modell över en salladsbar. Här visas kommunikation mellan de olika komponenterna, så som kommunikationen mellan AU och FCU, samt datan från AU till surfplattan. Även kommunikation mellan AU och dess databas syns.



Figur 3.3: Modell över picadelis back-end. Till vänster syns en webbserver, och till höger syns de objekt som markerats som “Out of scope” och som därför visas som gråa.

4. *Resultat*

I detta kapitel presenteras de hot som upptäckts via samtida litteratur samt från modelleringen i Threat Modeling Tool 2016. Efter att dessa hot presenteras kommer hotens relevans för Picadeli att diskuteras.

För den fullständiga rapporten från Threat Modeling Tool 2016, se bilaga A.

4.1 Hot mot sakernas internet

Enheter i ett sakernas internet är ovanligt sårbara, jämfört med persondatorer på grund av flera olika faktorer. Bland annat har enheterna ofta svaga lösenord, som snabbt går att gissa; saknar skydd såsom brandväggar och anti-virusprogram, då hårdvaran oftast begränsar prestandan; samt att säkerhetsuppdateringar installeras väldigt sällan eller inte alls [11].

Nedan presenteras några av de vanligaste typerna av hot mot enheter i ett sakernas internet. Hoten förekommer även mot persondatorer och andra typer av nätverksanslutna enheter, men är extra signifikanta inom sakernas internet.

4.1.1 Botnät

Ett vanligt förekommande problem inom sakernas internet är botnät, där enheter i ett sakernas internet som drabbats av skadlig mjukvara utför kommandon som utfärdats av en angripare. Dessa botnät utnyttjar sårbarheter och felaktigt konfigurerade enheter, och förökar sig autonomt med hjälp av dessa brister. Vidare är dessa nätverk decentraliserade för att maskera angriparens identitet, samt för extra redundans vid motåtgärder [11].

Det mest kända exemplet på ett sådant botnät är Mirai och variationer av detsamma. Genom att ta över sakernas internet-enheter — var för sig med låg kapacitet — har botnätet lyckats åstadkomma några av de kraftigaste överbelastningsattackerna hitintills genom den ackumulerade kapaciteten från samtliga enheter, vars antal uppskattats till runtomkring en halv miljon enheter [12].

4.1.2 Okrypterad fjärrförbindelse

Ett av de främsta problemen inom sakernas internet är osäkra kommunikationskanaler olika enheter emellan. Att just sakernas internet är osedvanligt utsatt beror på att kommunicerande enheter ofta är placerade på nätverk bortom en central organisations kontroll, vilket i sin tur medför att en illasinnad aktör som befinner sig på ett korsande nätverk eller kommunikationsmedium skulle kunna åstadkomma skada på data [13].

Kommunikation över osäkra medier kan — om inga eller otillräckliga åtgärder för att skydda datan har vidtagits — bland annat leda till att känsliga uppgifter görs tillgängliga för obehöriga samt att data manipuleras i ett eller båda håll medan kommunikationen äger rum.

4.1.3 Fysiska hot

Genom att inte ha fysisk kontroll på enheterna och miljön exponeras enheter inom sakernas internet även för hot på det fysiska planet. Här kan en angripare komma åt en enhet och utnyttja de möjligheter som ges av fysisk åtkomst. I en del fall har enheter hårdvarugränssnitt, så som JTAG. Genom att koppla upp sig mot dessa gränssnitt erbjuds ett felsökningsgränssnitt med vilket det är möjligt att läsa och redigera enhetens minne. Exempelvis skulle en enhet med JTAG i värsta fall kan leda till att känsliga konfigurationsfiler och krypteringsnycklar görs tillgängliga för angriparen [3].

Skydd mot dessa fysiska hot inkluderar mekanismer som rensar minnet när systemet upptäcker att det pågår ett fysiskt angrepp mot enheten. Vidare finns även diverse kryptografiska moduler som skyddar känsliga variabler i minnet [3].

4.2 Specifika hot mot Picadeli

Efter att ha studerat hot mot sakernas internet hölls ett möte med Picadeli. Vid det mötet bedömdes hoten utifrån vilka möjligheter de hade att utnyttjas för ett intrång givet Picadelis infrastruktur, och i detta kapitel kommer de hot som ska undersökas närmare att presenteras.

4.2.1 Autentisering

Då det vanligaste sättet för ett botnät att föröka sig, som diskuterats ovan, är genom att gissa lösenord, bestämdes det att en lösning för ett starkare skydd vid

detta skulle tas fram. Vid projektets inledning använde Picadeli lösenord på sakernas internet-enheterna för att fjärradministrera dem när så krävdes. Trots att inga angrepp uppdagats — autentiseringen är nämligen inte exponerad mot omvärlden — bestämdes det att förstärka autentiseringen genom att ersätta lösenordsinloggningen med en nyckelbaserad inloggning.

4.2.2 Kommunikation mellan VPN-klienter

För att förhindra spridning av skadlig mjukvara från en salladsbar till en annan, om ett intrång skulle inträffa, beslutades att en salladsbar endast ska ha tillgång till de tjänster som behövs för att den ska kunna utföra sin uppgift. Detta bedömdes vara möjligt att åstadkomma med hjälp av konfiguration av OpenVPN och en brandvägg — i det här fallet nftables. Motiveringen till detta är densamma som i fallet ovan. Om ett botnät försöker sprida sig genom de enheter en smittad enhet kommer åt i nätverket, kan spridning förhindras genom att inte tillåta kommunikation med fler enheter än nödvändigt.

4.2.3 Tillgänglighet

Ett problem som upptäcktes hade med tillgängligheten på salladsbarerna att göra. När en salladsbar sätts upp i en butik försöker den kommunicera med ett flertal tjänster på internet. Bland annat försöker den hämta tid och datum från en server på internet; hämta konfigurationsfiler för den specifika salladsbaren; samt hämta de reklamfilmer och produktbilder som ska visas på salladsbarens digitala skärmar. För att detta ska vara möjligt måste varje affär tillåta salladsbaren att kommunicera på de UDP- och TCP-portar som krävs — något som måste förhandlas fram med varje butiks IT-avdelning. För att minska beroendet på affärernas nätverkskonfiguration beslutades därför att all nätverkstrafik till och från en salladsbar ska gå genom en OpenVPN-server. På så sätt minskar beroendet av affärens nätverkskonfiguration, då allt som krävs är att affären tillåter trafik till och från VPN-servern.

Ett alternativt sätt att minska beroendet på affärernas nätverkskonfigurationer hade varit att använda sig av internetanslutningar över mobila 3G- eller 4G-nätverk, vilket även förekommer på en del salladsbarer. Detta medför däremot ett ökat administrationsbehov i form av hantering av SIM-kort och tillgänglig datamängd på varje SIM-kort.

4.3 Lösningar

Utifrån de hot som framkommit har ett antal lösningsförslag tagits fram. Lösningsförslagen är möjliga att tillämpa även för andra organisationer som bedömer att de

ovan beskrivna hoten kan utgöra en risk för dem.

4.3.1 Autentisering

För att konfigurera OpenSSH till att använda certifikat för inloggning krävs först och främst att en Certificate Authority (CA) är upprättad. Med hjälp av verktyget ssh-keygen som följer med varje installation av OpenSSH kan en användare generera ett nyckelpar som sedan används för assymetrisk kryptering. På den dator som ska agera CA genereras ett nyckelpar med hjälp av följande kommando:

```
ssh-keygen -t rsa -f ca_user.key
```

Här skapas ett nyckelpar, där den privata och publika nyckeln lagras i filerna ca_user.key respektive ca_user.key.pub. Nyckeltypen anges med flaggan -f, och anges här till en nyckel av RSA-typ.

När ett nyckelpar är skapat kan detta användas för att signera användares publika nycklar. Detta leder till att en certifikatsfil skapas, vilken en användare sedermera kan använda för att logga in med. I exemplet nedan signeras en användares publika nyckel id_rsa.pub med hjälp av CA:ns privata nyckel ca_user.key.

```
ssh-keygen -s ca_user.key -I foo -n bar -V +3650d
    id_rsa.pub
```

Vid körning av ovanstående kommando skapas en certifikatsfil, id_rsa-cert.pub, vilken tillåter inloggning som användaren bar, angivet i flaggan -n (för att tillåta inloggning som flera användare kan flera användare anges, separerade med kommatecken). Vidare ges certifikatet ett unikt ID med hjälp av flaggan -I — i detta fall foo.

På OpenSSH-servern behöver ändringar göras i konfigurationen för att tillåta inloggning med hjälp av certifikat. Genom att ange sökvägen till CA:ns publika nyckel tillåts inloggningar som signerats av CA:n. Detta sker med hjälp av konfigurationsdirektivet TrustedUserCAKeys i filen `/etc/ssh/sshd_config` på servern. Ett exempel ges nedan:

```
TrustedUserCAKeys       /etc/ssh/ca_user.pub
```

För att återkalla inloggning med ett visst certifikat måste konfigurationsdirektivet RevokedKeys anges i `/etc/ssh/sshd_config`, följt av sökvägen till en fil vars innehåll består av en lista med återkallade certifikat. Nedanstående exempel lägger till ett certifikat till listan över återkallade certifikat, givet dess unika ID foo:

```
1 echo "id:foo" > ./temp
2 ssh-keygen -k -u -f /etc/ssh/revoked_keys -s
    /etc/ssh/ca_user.pub ./temp
```

```
3 rm ./temp
```

Här återkallas certifikatet med hjälp av dess ID. För att detta ska vara möjligt måste en temporär fil skapas som innehåller ID:t som ska återkallas.

För användning inom sakernas internet gäller att denna lista skapas centralt, och distribueras till de olika enheterna på vilka inloggning ska ske. På så sätt undviks behovet av att generera denna lista separat på varje enskild enhet.

4.3.2 Kommunikation mellan VPN-klienter

För att säkerställa att endast behöriga VPN-klienter kan kommunicera med varandra har en lösning tagits fram, vilken består till hälften av konfiguration av en VPN-server (OpenVPN), och till hälften konfiguration av en brandvägg (nftables).

OpenVPN

För att separera sakernas internet-klienter från mänskliga klienter upprättades två stycken OpenVPN-instanser, på var sin UDP-port samt var sin virtuella nätverksenhet. För att tvinga klienterna att ansluta till rätt instans är klienternas nycklar uppdelade på två PKI:er.

Den första instansen, för klienter från sakernas internet, tilldelar klienterna en IP-adress från nätverket 10.10.0.0/16 eller fd77:12bb:f044:10::/64 — beroende på om klienten som ansluter använder en IPv4- eller IPv6-adress — på den virtuella nätverksenheten tun0. Den andra instansen, för mänskliga klienter, tilldelar klienten en IP-adress från 10.11.0.0/16 eller fd77:12bb:f044:11::/64 på den virtuella nätverksenheten tun1.

nftables

För att begränsa kommunikationen mellan obehöriga enheter ansågs nftables vara det bästa alternativet. Genom att, precis som sin föregångare iptables, använda sig av det i Linux-kärnan inbyggda ramverket Netfilter ges användaren möjligheter att filtrera nätverkstrafik på ett skräddarsytt vis. Vidare bearbetar nftables, till skillnad från föregångaren, enbart den nätverkstrafiken som är uttryckligen begärd, för att på så sätt minska på resursanvändningen på systemet.

Konfigurationsfilen inleds med ett direktiv för att rensa den för tillfället inlästa konfigurationen, följt av tre variabeldeklarationer.

```
1 flush ruleset
2
```

```
3 define nic_inet = eth0
4 define nic_saladbars = tun0
5 define nic_users = tun1
```

Listing 4.1: nftables konfigurationsfil, definitioner

På rad 29 i brandväggskonfiguration börjar definitionen av nätverksfiltret. En tabell (eng. table), *filter*, skapas, som i sin tur innehåller en kedja (eng. chain), *forward*. Denna kedja förses med ett antal regler som tillsammans definierar hur nätverkstrafik mellan olika OpenVPN-klienter får äga rum.

```
29 table inet filter {
30     chain forward {
31         type filter hook forward priority 0; policy
               accept;
32
33         # Block traffic from saladbars to other
               saladbars
34         oifname $nic_saladbars iifname
               $nic_saladbars drop
35
36         # Block traffic from saladbars to users
               unless communication
37         # is already established from a user. Also,
               block traffic
38         # between users.
39         oifname $nic_users ct state { new, invalid,
               untracked } drop
40         oifname $nic_users iifname $nic_users drop
41     }
42 }
```

Listing 4.2: nftables konfigurationsfil, filterregler

På rad 31 ges kedjans egenskaper. Här ges den typen *filter*, vilket möjliggör filtrerande av nätverkstrafiken. Vidare kopplas den till Netfilters *forward*-system, vilket innebär att den nätverkstrafik som bearbetas är trafik som passerar servern, utan att vare sig härstamma från eller vara ämnad för en lokal process. Slutligen anges *accept* som standardpolicy. Detta medför att all trafik är tillåten, såtillvida att den inte uttryckligen hindras i en av kedjans regler.

Den första regeln i kedjan presenteras på rad 34. Här förbjuds trafik som härstammar från en klient på salladsbarernas nätverksenhet och har densamma som mål. Eftersom kedjan är kopplad till *forward*-systemet får detta som följd att trafik mellan salladsbarer förbjuds, alltmedan trafik mellan en salladsbar och VPN-servern förblir opåverkad, vilket är nödvändigt för att VPN-servern och dess klienter ska kunna upprätthålla en förbindelse. På samma sätt som salladsbarer förhindras kom-

munikation, förhindras även mänskliga användare kommunikation på samma sätt på rad 40.

För att förbjuda trafik från salladsbarer till användare — utan att hindra salladsbarerna från att svara på en användares förfrågan — anges en tredje regel på rad 39. Denna förbjuder all trafik från en salladsbar till en användare, under förutsättningen att anslutningen inte redan är etablerad från en användare.

4.3.3 Tillgänglighet

Genom en kombination av OpenVPN och nftables togs en lösning fram för att tunnla all trafik från en klient genom servern. Här kommer konfigurationsfilen både för OpenVPN och nftables att presenteras och förklaras.

Två konfigurationsfiler för OpenVPN togs fram — en för salladsbarer och en för användare — men då skillnaderna är försumbara kommer endast den förstnämnda att presenteras här.

För att minimera datamängden som skickas mellan klienter och servern instrueras servern att komprimera datan med hjälp av LZ4, samt att meddela detta till de anslutande klienterna (rad 5 och 6). Av de komprimeringsalgoritmer som stöds av OpenPVN — alternativen är LZO och LZ4 — är LZ4 den algoritm som förbrukar minst resurser [6].

För kryptering av datatrafiken används AES-algoritmen, med en nyckellängd på 256 bitar i läget Cipher Block Chaining (CBC). Se rad 11. AES såväl som CBC är beprövade tekniker som erbjuder ett bra skydd för den krypterade datan.

På rad 9-12 anges bland annat de filer som används för att kryptera den trafik som går över datakanalen i OpenVPN. Även filer för att autentisera användare anges här.

Från rad 19 till och med rad 23 anges ett antal direktiv som används för att sänka privilegiet på OpenVPN-processen till det minsta möjliga. Detta görs för att minimera de skador som skulle kunna uppstå om en sårbarhet i OpenVPN upptäcks och utnyttjas av en angripare, för att begränsa åtkomsten till känslig data.

```
1  port 1194
2  proto udp
3  dev tun0
4  keepalive 50 150
5  compress lz4
6  push "compress lz4"
7  topology subnet
8
9  ca /etc/openvpn/easy-rsa/keys/ca.crt
10 cert /etc/openvpn/easy-rsa/keys/server.crt
```

```
11 key /etc/openvpn/easy-rsa/keys/server.key
12 dh /etc/openvpn/easy-rsa/keys/dh2048.pem
13 cipher AES-256-CBC    # AES
14
15 server 10.10.0.0 255.255.0.0
16 server-ipv6 fd77:12bb:f044:10::/64
17 push "redirect-gateway def1 ipv6"
18
19 user nobody
20 group nogroup
21 ifconfig-pool-persist ipp_saladbars.txt
22 persist-key
23 persist-tun
24
25 status openvpn-status.log
26 verb 3
```

Listing 4.3: OpenVPN:s konfigurationsfil

I nftables konfigurationsfil förbereds servern för att tunnla trafiken från klienter. Genom att använda sig av Network Address Translation (NAT) kommer varje VPN-klient att erhålla VPN-serverns publika IP-adress när den kommunicerar över internet. Genom att upprätthålla en NAT-tabell kan VPN-servern skicka svaret på trafiken tillbaka till rätt VPN-klient så fort det erhållits.

I sin nuvarande version tillåter inte nftables en enad konfiguration av NAT för IPv4 och IPv6 än [14], så därför konfigureras var sin tabell i brandväggen för IPv4 och IPv6 — *nat* på rad 7 respektive *nat6* på rad 18. För att aktivera NAT på den utgående trafiken registreras en kedja, döpt *postrouting* i båda tabellerna. Denna är av typen *nat* och kopplas till netfilters *postrouting*-hook. Kedjorna innehåller en enda regel, som sätter källans IP-adress till den på det specificerade nätverkskortet. För att returtrafiken ska vidarebefodras till rätt klient måste även en kedja, *prerouting*, registreras på netfilter-hooken *prerouting*, även om den inte innehåller någon regel.

```
 7 table ip nat {
 8         chain prerouting {
 9                 type nat hook prerouting priority
                      0; policy accept;
10         }
11
12         chain postrouting {
13                 type nat hook postrouting priority
                      0; policy accept;
14                 oifname $nic_inet masquerade
15         }
16 }
17
```

```
18 table ip6 nat6 {
19         chain prerouting {
20                 type nat hook prerouting priority
                      0; policy accept;
21         }
22
23         chain postrouting {
24                 type nat hook postrouting priority
                      0; policy accept;
25                 oifname $nic_inet masquerade
26         }
27 }
```

Listing 4.4: nftables konfigurationsfil, NAT-regler

5. *Lärdomar från litteratur- och fallstudien*

I detta kapitel kommer de lärdomar och slutsatser som går att härleda till fallstudien och den inhämtade litteraturen att diskuteras. Först kommer lämpligheten i att använda Microsoft Threat Modeling Tool 2016 för att utvärdera säkerheten inom sakernas internet att diskuteras, samt vad som skiljer detta verktyg från ett idealt sådant. Därefter kommer arkitekturella frågor att lyftas. Här diskuteras viktiga designval att diskuteras som minimerar riskerna för ett angrepp. Till slut avslutas kapitlet med en diskussion om de rutiner som är nödvändiga för att förhindra angrepp på ett system som använder sig av sakernas internet.

5.1 Threat Modeling Tool 2016

Över lag fungerar Threat Modeling Tool 2016 rätt bra för att utvärdera säkerheten inom sakernas internet, trots att det inte är det programmet är framtaget för att göra.

En av de funktionaliteter som Threat Modeling Tool 2016 saknar är förmågan att gruppera olika processer och tjänster som körs på samma enhet som just en enhet. Av den anledningen rapporteras många falska alarm, speciellt i de sammanhang då databaser är inblandade. Anledningen till detta är att verktyget förutsätter att en angripare, på ett eller annat sätt, placera sig mellan exempelvis en webbserver och en databas för att tjuvlyssna på eller manipulera trafiken, när de båda tjänsterna i själva verket körs på en och samma enhet.

Tack vare de många medföljande standardobjekten fungerar verktyget trots allt bra för att modellera sakernas internet. Både objekten för webbservrar, databaser, webbapplikationer och dyilkt såväl som anslutningar mellan dessa objekt erbjuder många konfigurationsmöjligheter för att anpassa modellen efter det modellerade systemet. Skulle det krävas någon form av specialobjekt finns dessutom möjligheten för att skapa ett sådant, komplett med regler för vad som bör anses vara en potentiell sårbarhet.

5.2 Arkitektur

En av de största riskerna när det kommer till datakonfidentialitet är ett undermåligt eller icke-existerande skydd för data som skickas från en enhet till en annan [13]. Av den anledningen är det viktigt att använda sig av transportkryptering då data ska skickas mellan enheter, för att förhindra att en angripare kan komma över trafik i klartext.

För trafik som går till en webbläsare rekommenderas att använda HTTPS istället för HTTP. Genom användandet av TLS i HTTPS ges både sändare och mottagare skydd mot avlyssning och manipulerad data.

För övrig trafik där protokollen i sig inte tillåter krypterad data rekommenderas att känslig data skickas över ett VPN med krypterade anslutning. Genom att skicka trafik genom en sådan nätverkstunnel kommer trafiken att krypteras oavsett datans tillstånd då den skickas från en process på enheten.

För de organisationer som använder sig sakernas internet i deras egna verksamhet rekommenderas dessutom att en central server sätts upp dit alla loggar skickas. Om en angripare tillskansar sig fulla rättigheter på en enhet kan angriparen stänga av loggning samt rensa de som redan finns på enheten. Genom att använda sig av central loggning kan en organisation åtminstone rekonstruera delar av händelseförloppet.

5.3 Rutiner

Genom att kontinuerligt granska loggfiler ges en administratör möjligheten att upptäcka förändringar i hotbilden mot en enhet eller ett system och agera därefter.

Vidare är det viktigt att en administratör håller sig uppdaterad vad gäller nyligen upptäckta sårbarheter. Skulle det upptäckas en sårbarhet i en mjukvara som en organisation använder sig av, är det en risk att inte åtgärda problemet innan en angripare hinner utnyttja det.

I enlighet med ovanstående paragraf är det av intresse att uppdatera de mjukvaror som används inom organisationen, då de ofta täpper till sårbarheter som upptäckts. Om det inte finns en uppdatering, eller om det av någon anledning inte är möjligt att uppdatera nuvarande konfiguration är det viktigt att åtminstone kringgå problemet genom att exempelvis blockera tillgång till den sårbara tjänsten för de som kan tänkas utnyttja sårbarheten.

5.4 Återhämtningsplan

Om ett intrång skulle äga rum är det viktigt att ha en återhämtningsplan. Hanteringen av en sådan kan delas in i fyra steg: planering; upptäckt av incident och en förståelse för händelseförloppet; karantänisering av utsatta system för att återställa detsamma; samt efterföljande incidenthantering i form av loggföring och liknande [3].

I planeringsstadiet gäller det att planera olika scenarion. Vad ska göras om alla enheter tappar kontakten med varandra eller med den centrala servern? Vem ska kontaktas om en angripare kommer åt känsliga uppgifter och försöker utpressa organisationen på pengar? Hur hanteras enheter på sakernas internet som leder till att människor eller egendom skadas?

För att upptäcka ett pågående intrång rekommenderas att använda ett security information and event management-program (SIEM-program). Genom att analysera och korrelera bland annat loggfiler, datatrafik, och rapporter från antivirusprogram och Intrusion Detection System (IDS) kan en incident upptäckas och rapporteras [15].

När ett intrång väl har upptäckts gäller det att identifiera var sökandet efter bevis ska äga rum. En försvårande omständighet vad gäller sakernas internet är att enheter inte nödvändigtvis befinner sig i ett och samma nätverk hela tiden. I de fall då en mobil enhet, så som en surfplatta eller pekskärmstelefon kopplas upp mot ett sakernas internet kan vilket nätverk enheten varit uppkopplad mot vara av juridisk betydelse för utredningen. Om en enhet som är inblandad i en incident varit uppkopplad från en organisations kontor är det lättare att få tillgång till data från kringliggande enheter (exempelvis loggfiler från en router) än om enheten varit uppkopplad från nätverket i ett hushåll [16]. Notera att detta särskilt gäller för organisationer där enheter kan befinna sig i olika länder.

För att begränsa skadan gäller det att sätta smittade enheter i karantän. Genom att flytta en smittad enhet till en testmiljö, eller att dirigera om trafiken till ett testnätverk, är det möjligt att fortsatt studera beteendet hos den smittade enheten, och i förlängningen även angriparens avsikter. Vad gäller de behov för att ersätta den smittade enheten gäller det att ha en ren diskavbild som går att föra över till den enhet som är tänkt att ersätta den gamla [3].

Efter att en incident upptäckts och avhjälpts återstår att granska tillståndet hos övriga enheter för att försäkra sig om att problemet lösts. Utöver det måste den ursprungliga orsaken till intrånget åtgärdas för att förhindra att det upprepas [3].

5.5 Etik

Verktyg såsom Threat Modeling Tool 2016 från Microsoft är ett väldigt bra verktyg för att utvärdera säkerheten i ett system. Skulle en sårbarhet upptäckas, som låter en angripare att få otillbörlig tillgång till systemet är det viktigt att hantera denna kunskap med varsamhet — och framförallt att inte utnyttja kunskapen själv med fientliga avsikter.

I Sverige är dataintrång ett brott som kan leda till fängelse i högst sex år i de fall då brottet anses vara grovt [17]. Vidare kan ett otillåtet utnyttjande av sårbarheter leda till andra brottsrubricering om en angripare skulle tillskansa sig exempelvis pengar, personuppgifter eller dylikt genom intrånget.

6. *Fortsatt arbete*

Något som ej undersökts är skydd mot fysiska angrepp på enheter. Vid en fortsättningsstudie skulle det därför vara intressant att studera vilka typer av skydd som finns för enheter på sakernas internet, samt dess för- och nackdelar.

Även så kallad social manipulation (eng. social engineering), som i viss mån relaterar till ovanstående hade varit av intresse för vidare studier. Hur kan exempelvis en organisation skydda sig mot någon som utger sig för att representera organisationen? Som ett exempel kan tänkas ett företag som ägnar sig åt övervakning av nätverkstrafik på ett sjukhus. Om en utomstående kan besöka sjukhuset, och utan att identifiera sig få fysisk tillgång till den hårdvara som företaget placerat på sjukhuset, hade det i värsta fall kunnat få konsekvenser för patienter och anställda på sjukhuset.

7. *Litteraturförteckning*

[1] IEEE. (2015) Towards a definition of the Internet of Things (IoT). [Online]. Tillgänglig: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf, hämtad: 2018-03-05.

[2] J. Huang and K. Hua, *Managing the Internet of Things - Architectures, Theories and Applications*. Institution of Engineering and Technology, 2017. [Online]. Tillgänglig: https://app.knovel.com/hotlink/toc/id:kpMITATA02/managing-internet-things/managing-internet-things, hämtad: 2018-04-30.

[3] B. Russell and D. van Duren, *Practical Internet of Things Security - A Practical, Indispensable Security Guide That Will Navigate You through the Complex Realm of Securely Building and Deploying Systems in Our IoT-Connected World*. Packt Publishing, 2016. [Online]. Tillgänglig: https://app.knovel.com/hotlink/toc/id:kpPITSAPIA/practical-internet-things/practical-internet-things, hämtad: 2018-04-02.

[4] T. Ylönen. (2006) The Secure Shell (SSH) Protocol Architecture. [Online]. Tillgänglig: https://tools.ietf.org/html/rfc4251, hämtad: 2018-03-17.

[5] T. Ylönen. (2006) The Secure Shell (SSH) Authentication Protocol. [Online]. Tillgänglig: https://tools.ietf.org/html/rfc4252, hämtad: 2018-03-17.

[6] OpenVPN. openvpn - secure IP tunnel daemon. [Online]. Tillgänglig: https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage, hämtad: 2018-05-29.

[7] OpenVPN. Community Project Overview. [Online]. Tillgänglig: https://openvpn.net/index.php/open-source/245-community-open-source-software-overview.html, hämtad: 2018-04-10.

[8] OpenVPN. Why SSL VPN? [Online]. Tillgänglig: https://openvpn.net/index.php/open-source/339-why-ssl-vpn.html, hämtad: 2018-04-10.

[9] H. Gantenbein. (2016) STRIDE, CIA and the Modern Adversary. [Online]. Tillgänglig: https://blogs.msdn.microsoft.com/heinrichg/2016/06/07/stride-cia-and-the-modern-adversary/, hämtad: 2018-04-07.

[10] Microsoft. SDL Threat Modeling Tool. [Online]. Tillgänglig: https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx, hämtad: 2018-04-06.

[11] E. Bertino and N. Islam, "Botnets and Internet of Things Security," *Computer*, vol. 50, no. 2, pp. 76–79, February 2017.

[12] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDos in the IoT: Mirai and Other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, July 2017.

[13] S. Li and L. D. Xu, *Securing the Internet of Things*. Syngress Publishing, 2017. [Online]. Tillgänglig: http://library.books24x7.com.proxy.lib.chalmers.se/toc.aspx?site=Y7V97&bookid=123329, hämtad: 2018-04-13.

[14] A. B. Gonzalez. Performing Network Address Translation (NAT). [Online]. Tillgänglig: https://wiki.nftables.org/wiki-nftables/index.php/Performing_Network_Address_Translation_(NAT), hämtad: 2018-05-15.

[15] B. N, "Security information and event management (siem) – a detailed explanation." [Online]. Tillgänglig: https://gbhackers.com/security-information-and-event-management-siem-a-detailed-explanation/, hämtad: 2018-06-03.

[16] E. Oriwoh, D. Jazani, G. Epiphaniou, and P. Sant, "Internet of Things Forensics: Challenges and Approaches," in *9th International Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*, 10 2013, pp. 608–615. [Online]. Tillgänglig: https://www.researchgate.net/publication/259332114_Internet_of_Things_Forensics_Challenges_and_Approaches

[17] Svensk författningssamling, "Brottsbalk (1962:700)." [Online]. Tillgänglig: http://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/brottsbalk-1962700_sfs-1962-700, hämtad: 2018-06-11.

A. *Threat Modeling Tool 2016*

Threat Modeling Report

Created on 6/1/2018 1:59:35 AM

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

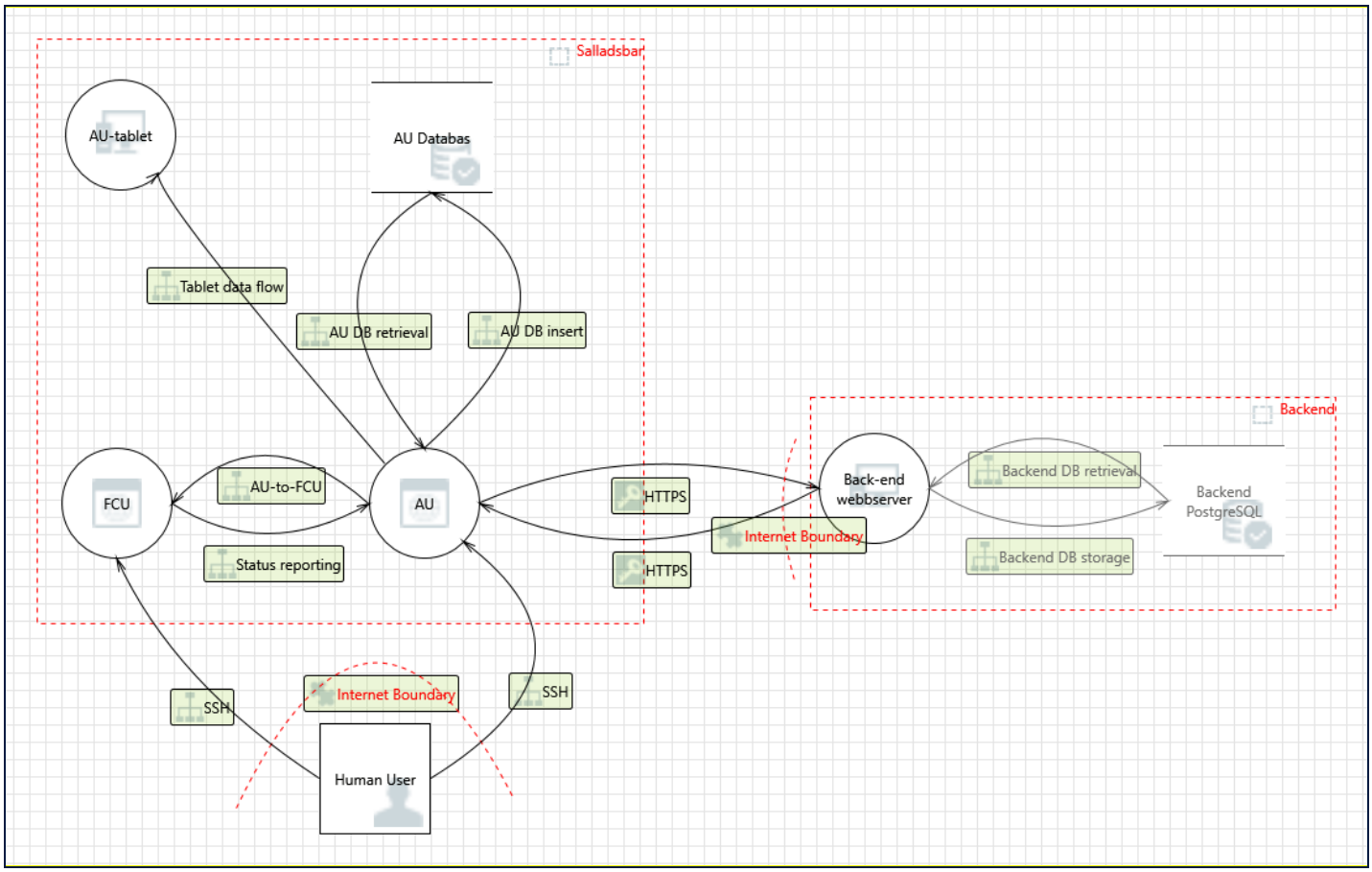
Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	0
Not Applicable	28
Needs Investigation	14
Mitigation Implemented	13
Total	55
Total Migrated	0

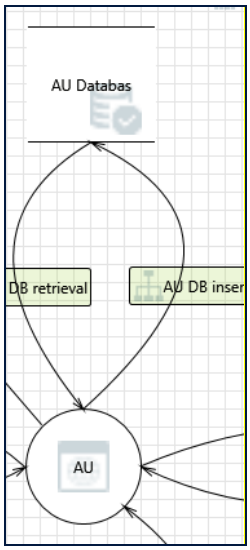
Diagram: Picadeli



Picadeli Diagram Summary:

Not Started	0
Not Applicable	28
Needs Investigation	14
Mitigation Implemented	13
Total	55
Total Migrated	0

Interaction: AU DB insert



1. Potential Excessive Resource Consumption for AU or SQL Database [State: Mitigation Implemented] [Priority: High]

Category: Denial Of Service
Description: Does AU or AU PostgreSQL take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.
Justification: Optimerade SQL-frågor -> låg resursförbrukning.

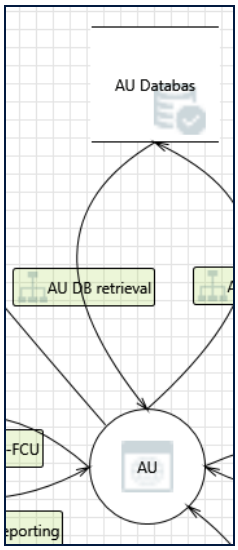
2. Potential SQL Injection Vulnerability for SQL Database [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.
Justification: SQL-frågor städas innan de når databasen.

3. Spoofing of Destination Data Store SQL Database [State: Not Applicable] [Priority: High]

Category: Spoofing
Description: AU PostgreSQL may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of AU PostgreSQL. Consider using a standard authentication mechanism to identify the destination data store.
Justification: Databas på samma enhet som AU-mjukvaran.

Interaction: AU DB retrieval



4. Weak Access Control for a Resource [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure
Description: Improper data protection of AU PostgreSQL can allow an attacker to read information not intended for disclosure. Review authorization settings.
Justification: Data endas tillgänglig via localhost.

5. Persistent Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'AU' could be a subject to a persistent cross-site scripting attack because it does not sanitize data store 'AU PostgreSQL' inputs and output.
Justification: Data är städad.

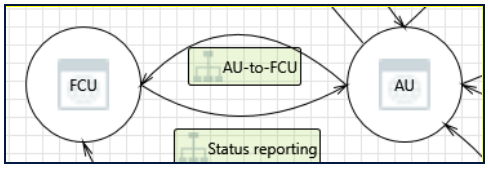
6. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'AU' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: Indata städas där det behövs.

7. Spoofing of Source Data Store SQL Database [State: Not Applicable] [Priority: High]

Category: Spoofing
Description: AU PostgreSQL may be spoofed by an attacker and this may lead to incorrect data delivered to AU. Consider using a standard authentication mechanism to identify the source data store.
Justification: Databas befinner sig på samma enhet som AU-mjukvaran.

Interaction: AU-to-FCU



8. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: FCU may be able to impersonate the context of AU in order to gain additional privilege.
Justification: Finns inga extra privilegier att tillskansa.

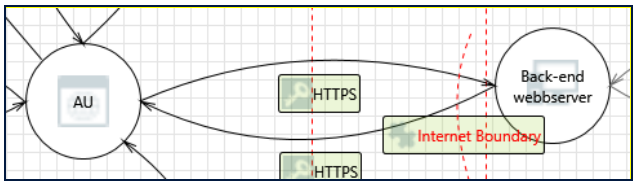
9. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'FCU' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: Data städas.

10. AU Process Memory Tampered [State: Not Applicable] [Priority: High]

Category: Tampering
Description: If AU is given access to memory, such as shared memory or pointers, or is given the ability to control what FCU executes (for example, passing back a function pointer.), then AU can tamper with FCU. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
Justification: AU- och FCU-mjukvaran körs inte på samma enhet, och har därmed inte tillgång till ett gemensamt minne.

Interaction: HTTPS



11. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'cloud.picadeli.com' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: Data städas.

12. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: cloud.picadeli.com may be able to impersonate the context of AU in order to gain additional

privilege.

Justification: Finns inga ytterligare privilegier att tillskansa.

13. Spoofing the AU Process [State: Needs Investigation] [Priority: High]

Category: Spoofing

Description: AU may be spoofed by an attacker and this may lead to unauthorized access to cloud.picadeli.com. Consider using a standard authentication mechanism to identify the source process.

Justification: <no mitigation provided>

14. Potential Data Repudiation by cloud.picadeli.com [State: Not Applicable] [Priority: High]

Category: Repudiation

Description: cloud.picadeli.com claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: All data sparas.

15. Potential Process Crash or Stop for cloud.picadeli.com [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: cloud.picadeli.com crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: <no mitigation provided>

16. Data Flow HTTPS Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>

17. cloud.picadeli.com May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: AU may be able to remotely execute code for cloud.picadeli.com.

Justification: Finns inget stöd för detta i mjukvaran.

18. Elevation by Changing the Execution Flow in cloud.picadeli.com [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into cloud.picadeli.com in order to change the flow of program execution within cloud.picadeli.com to the attacker's choosing.

Justification: Data städas.

19. Cross Site Request Forgery [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting, … The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations.

Justification: Webbläsare används ej.

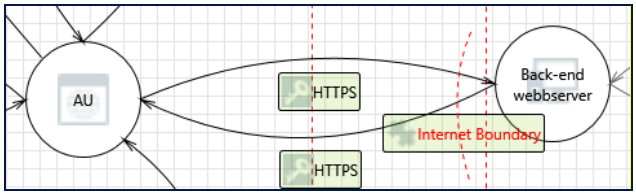
20. JavaScript Object Notation Processing [State: Not Applicable] [Priority: High]

Category: Tampering

Description: If a dataflow contains JSON, JSON processing and hijacking threats may be exploited.

Justification: Tampering inte möjligt p.g.a. SSH.

Interaction: HTTPS



21. cloud.picadeli.com Process Memory Tampered [State: Not Applicable] [Priority: High]

Category: Tampering
Description: If cloud.picadeli.com is given access to memory, such as shared memory or pointers, or is given the ability to control what AU executes (for example, passing back a function pointer.), then cloud.picadeli.com can tamper with AU. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
Justification: Separata enheter -> separata minnen.

22. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'AU' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: Data städas.

23. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: AU may be able to impersonate the context of cloud.picadeli.com in order to gain additional privilege.
Justification: Ej möjligt.

24. Potential Data Repudiation by AU [State: Needs Investigation] [Priority: High]

Category: Repudiation
Description: AU claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

25. Potential Process Crash or Stop for AU [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: AU crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: <no mitigation provided>

26. Data Flow HTTPS Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

27. AU May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: cloud.picadeli.com may be able to remotely execute code for AU.
Justification: Finns inget stöd för det i mjukvaran.

28. Elevation by Changing the Execution Flow in AU [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into AU in order to change the flow of program execution within AU to the attacker's choosing.
Justification: Data städas.

29. Collision Attacks [State: Not Applicable] [Priority: High]

Category: Tampering
Description: Attackers who can send a series of packets or messages may be able to overlap data. For example, packet 1 may be 100 bytes starting at offset 0. Packet 2 may be 100 bytes starting at offset 25. Packet 2 will overwrite 75 bytes of packet 1. Ensure you reassemble data before filtering it, and ensure you explicitly handle these sorts of cases.
Justification: <no mitigation provided>

30. Weak Authentication Scheme [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure
Description: Custom authentication schemes are susceptible to common weaknesses such as weak credential change management, credential equivalence, easily guessable credentials, null credentials, downgrade authentication or a weak credential change management system. Consider the impact and potential mitigations for your custom authentication scheme.
Justification: Välbeprövade autentiseringsmekanismer används.

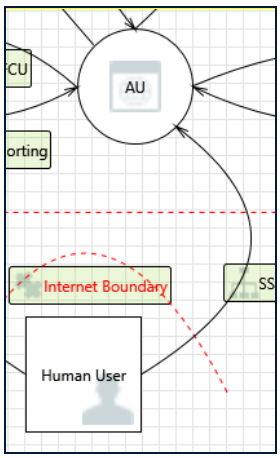
31. Replay Attacks [State: Not Applicable] [Priority: High]

Category: Tampering
Description: Packets or messages without sequence numbers or timestamps can be captured and replayed in a wide variety of ways. Implement or utilize an existing communication protocol that supports anti-replay techniques (investigate sequence numbers before timers) and strong integrity.
Justification: Skyddat m.h.a. HTTPS.

32. JavaScript Object Notation Processing [State: Not Applicable] [Priority: High]

Category: Tampering
Description: If a dataflow contains JSON, JSON processing and hijacking threats may be exploited.
Justification: Tampering inte möjligt p.g.a. SSH.

Interaction: SSH



33. Spoofing the AU Process [State: Needs Investigation] [Priority: High]

Category: Spoofing
Description: AU may be spoofed by an attacker and this may lead to information disclosure by Human User. Consider using a standard authentication mechanism to identify the destination process.
Justification: <no mitigation provided>

34. Cross Site Scripting [State: Not Applicable] [Priority: High]

Category: Tampering
Description: The web server 'AU' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: All kommunikation här sker m.h.a. SSH.

35. Potential Data Repudiation by AU [State: Needs Investigation] [Priority: High]

Category: Repudiation
Description: AU claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

36. Potential Process Crash or Stop for AU [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: AU crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: <no mitigation provided>

37. Data Flow Generic Data Flow Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>

38. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: AU may be able to impersonate the context of Human User in order to gain additional privilege.
Justification: Enkelriktad kommunikation.

39. AU May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: Human User may be able to remotely execute code for AU.
Justification: Detta är poängen med kommunikationen.

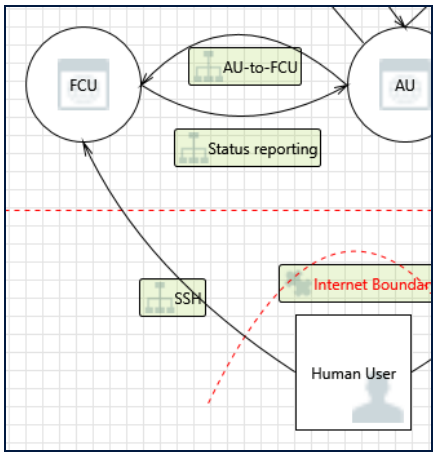
40. Elevation by Changing the Execution Flow in AU [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into AU in order to change the flow of program execution within AU to the attacker's choosing.
Justification: Obehöriga utestängda tack vare nyckelbaserad inloggning.

41. Cross Site Request Forgery [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting, … The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations.
Justification: Endast kommunikation över SSH.

Interaction: SSH



42. Cross Site Request Forgery [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting, … The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations.
Justification: Endast kommunikation över SSH.

43. Elevation by Changing the Execution Flow in FCU [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into FCU in order to change the flow of program execution within FCU to the attacker's choosing.
Justification: Inloggning m.h.a. nyckelpar förhindrar detta.

44. FCU May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: Human User may be able to remotely execute code for FCU.
Justification: Denna kommunikation sker just med den avsikten.

45. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: FCU may be able to impersonate the context of Human User in order to gain additional privilege.
Justification: Finns inga privilegier att utöka p.g.a. enkelriktad kommunikation.

46. Data Flow Generic Data Flow Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

47. Potential Process Crash or Stop for FCU [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: FCU crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: <no mitigation provided>

48. Potential Data Repudiation by FCU [State: Needs Investigation] [Priority: High]

Category: Repudiation
Description: FCU claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification: <no mitigation provided>

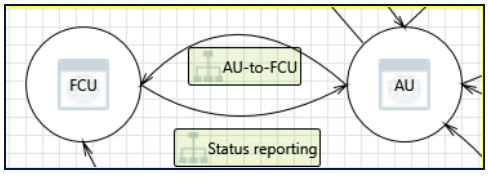
49. Cross Site Scripting [State: Not Applicable] [Priority: High]

Category: Tampering
Description: The web server 'FCU' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: Ingen trafik av detta slag mellan enheterna.

50. Spoofing the FCU Process [State: Needs Investigation] [Priority: High]

Category: Spoofing
Description: FCU may be spoofed by an attacker and this may lead to information disclosure by Human User. Consider using a standard authentication mechanism to identify the destination process.
Justification: <no mitigation provided>

Interaction: Status reporting



51. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege
Description: AU may be able to impersonate the context of FCU in order to gain additional privilege.
Justification: Finns inga extra privilegier att tillskansa.

52. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'AU' could be a subject to a cross-site scripting attack because it does not sanitize untrusted

input.

Justification: Datan städas.

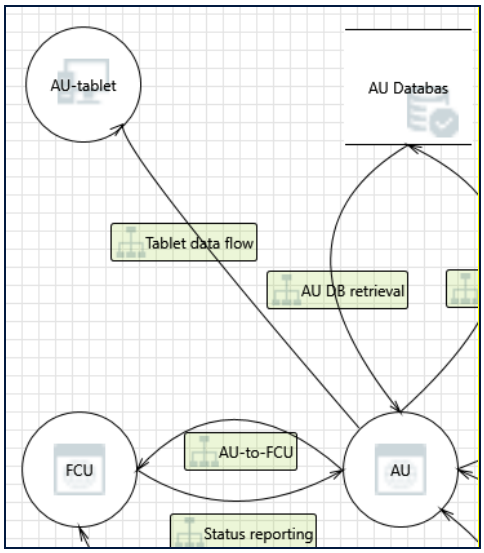
53. FCU Process Memory Tampered [State: Not Applicable] [Priority: High]

Category: Tampering

Description: If FCU is given access to memory, such as shared memory or pointers, or is given the ability to control what AU executes (for example, passing back a function pointer.), then FCU can tamper with AU. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.

Justification: AU- och FCU-mjukvarorna körs inte på samma enhet, och har därför inte tillgång till varandras minnen.

Interaction: Tablet data flow



54. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: AU-tablet may be able to impersonate the context of AU in order to gain additional privilege.

Justification: Finns inga extra privilegier att tillskansa.

55. Cross Site Scripting [State: Not Applicable] [Priority: High]

Category: Tampering

Description: The web server 'AU-tablet' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.

Justification: Källan skickar alltid kontrollerad data.