NACKADEMIN

2023-02-18
IOT22

Individuell inlämning

- Detect Device Tampering with Microsoft Defender for IoT

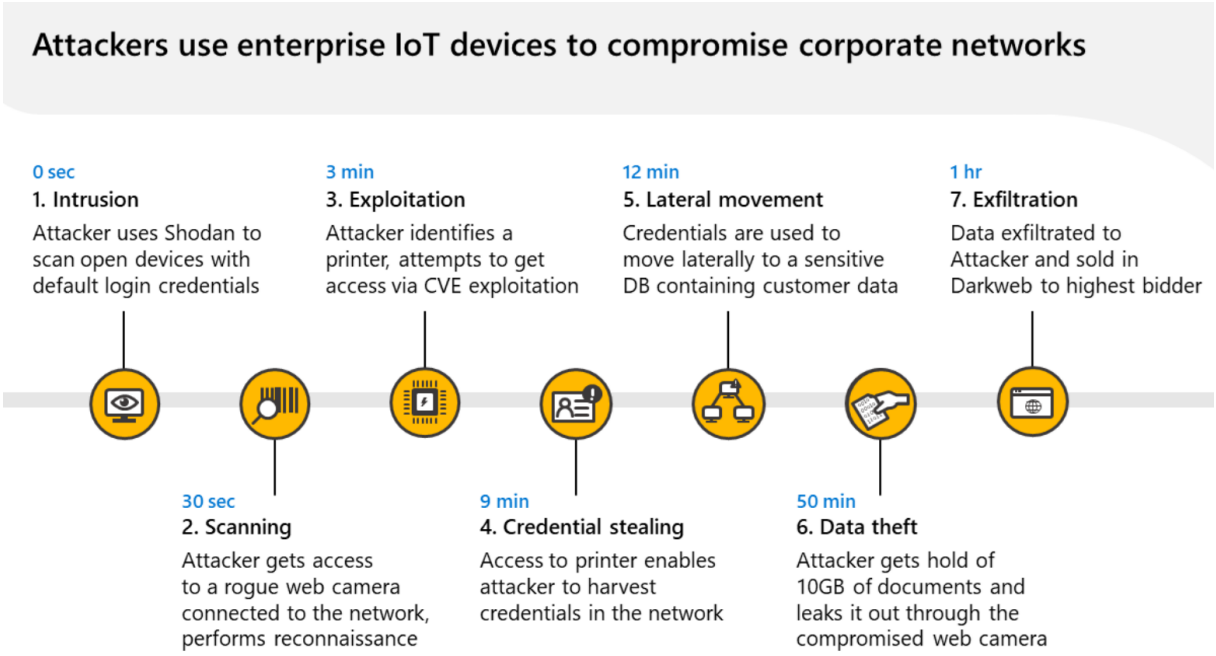
Namn: Martin Myrberg
E-post: Martin.Myrberg@yh.nackademin.se

Innehållsförteckning

Inledning

Användning av internet of things (IoT) är idag ett vanligt förekommande inslag hos många företag, inte minst inom den industriella sektorn. IoT fungerar som ett verktyg för att få en ökad insikt i den operativa verksamheten och därigenom förbättrad kontroll och beslutsunderlag. IoT-enheter kan också bidra till ökad effektivitet och ger möjlighet till flertalet användningsområden där sensordata kan ge betydelsefull information. Som följd av en bättre insyn kommer dock också en större exponeringsyta att utsättas för cyberattacker. IoT-enheter saknar ofta samma rutinmässiga skyddsmekanismer som finns i till exempel bärbara datorer och mobiltelefoner vilket gör dem mer sårbara för potentiella intrång. Det är alltså inte ovanligt att illasinnade aktörer använder sig av dessa enheter som en ingångspunkt för att sedan komma vidare in i nätverket och stjäla information eller orsaka skada.

Nedan bild illustrerar ett exempel på detta och involverar två IoT-enheter, där den ena används för att ta sig in på nätverket och den andra för vidare förflyttning till att slutligen stjäla information.



För att hantera de stora datamängder som ett nätverk av IoT-enheter genererar och samtidigt skydda sig mot attacker, kopplas ofta enheterna upp mot en molntjänst. Genom att koppla till molnet är det enkelt att skala upp sin IoT-lösning och samtidigt hålla nere arbetsbelastning (kostnader) för utveckling, drift, administration, och underhåll. Ett exempel på molntjänst som används i detta sammanhang är Microsoft Azure. Azure erbjuder ett brett utbud av tjänster med alltifrån databaser till maskininlärning och databearbetning för IoT-lösningar.

Denna rapport syftar till att gå närmare in på säkerhetsaspekterna av just Azure som IoT-plattform och tar sin utgångspunkt i en praktisk tillämpning av Microsofts egna lärolabb *”Detect Device Tampering with Microsoft Defender for IoT”* för att besvara följande frågeställning:

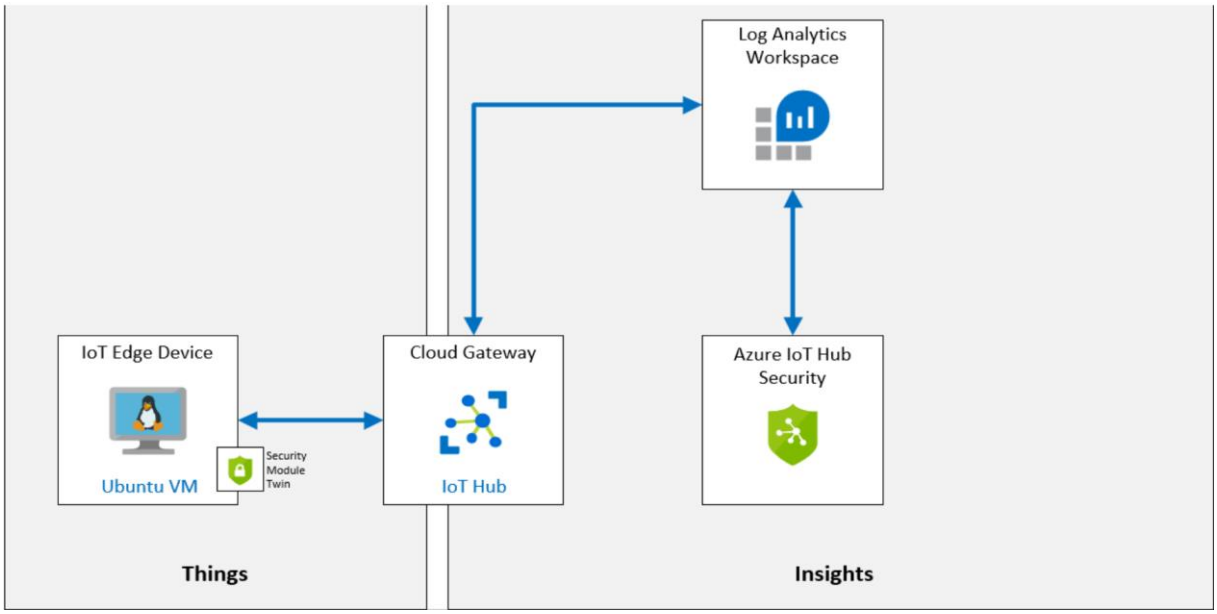
”Vilka områden i en IoT-lösning står inför sårbarheter och hur kunde Microsoft Defender avvärja dessa under labbens gång?”

Scenario

Labben utgår från ett fiktivt företag kallat ”Contoso” vilka tillverkar ost. För att hantera en ökad efterfrågan har de bland annat fått en ny produktionslina, utrustad med IoT-enheter, som ska hjälpa till med packningen av nya order. Vi får i uppdrag att se till att dessa nya enheter är säkra samt få tillgång till säkerhetsrekommendationer genom användning av Microsoft Defender for IoT. Contoso har dessutom installerat nya termostater i deras lager i syfte att förbättra kontrollen av temperaturen. Här får vi i uppgift att skapa anpassade larm/varningsnotiser för att övervaka om termostaterna överskrider den förväntade överföringsfrekvensen av telemetri.

IoT-arkitektur

Nedan bild visar de resurser, med tillhörande dataflöde, vi blir instruerade att sätta upp i labben.



IoT-hubben är central i denna lösning och fungerar som ett nav för datatrafiken mellan IoT-enheterna och dess applikationer. I detta fall samlar hubben in data från enheterna, som simuleras i labben med en virtuell maskin, behandlar informationen och skickar den sedan vidare till Log Analytics Workspace. Denna ”Workspace” skapas och synkas med Microsoft Defender vilket gör det möjligt för oss att se eventuella säkerhetslarm, händelser och rekommendationer för våra IoT-enheter som vi sedan kan välja att vidta åtgärder på.

Sårbarheter och Säkerhetsåtgärder

Under laborationens gång stöter vi på olika moment där sårbarheter i vår lösning blottar sig. Initialt, när vi sätter upp vår virtuella maskin för att simulera IoT-enheterna, använder vi oss av SSH-protokollet för att ansluta till dessa. SSH är vanligt förekommande för kommunikation med IoT-enheter och anses förvisso säkert. Däremot blir vi instruerade att välja lösenord som autentiseringsmetod istället för SSH-nyckelpar vilket ökar risken för obehörig åtkomst och sårbarheten för t.ex. ”man-in-the-middle-attacker”. Visserligen finns alternativet i Azure att använda SSH-nycklar men med tanke på att vi instrueras att välja lösenord i labben tas det ändå upp som en potentiell sårbarhet.

Innan vi kan ansluta våra enheter till IoT-hubben behöver de registreras. Vid registrering får man möjlighet att göra ett val av krypteringsmetod för autentisering; symmetriska nycklar eller två varianter av X.509 certifikat. Här instrueras vi att välja symmetriska nycklar vilket är relativt smidigt då det endast kräver en nyckel för att kryptera och dekryptera informationen. Däremot finns ingen mekanism hos symmetriska nycklar för att verifiera identiteten på de kommunicerande parterna vilket orsakar sårbarhet vid t.ex. en ”burte force-attack”. En potentiell lösning för detta skulle kunna vara att kombinera SSH-nyckelpar (för att säkra autentiseringen) med symmetriska nycklar (för att stärka krypteringen) i kommunikationen mellan vår IoT-enhet och hubb.

Vidare i labben får vi i uppdrag att skapa en ”Security Module Twin” (SMT). SMT är en digital tvilling av IoT-enhetens säkerhetmodul och innehåller all relevant information för enhetens säkerhet. Men eftersom tvillingen delar viktig information med det underliggande systemet innebär det också att den kan bli en potentiell källa för dataintrång. Framförallt kan felaktig konfiguration av tvillingen göra vår IoT-enhet sårbar och öka risken för till exempel ”side-channel-attacker”, där en angripare kommer in via vår SMT som en alternativ väg in i vår IoT-lösning.

För att addera ytterligare ett säkerhetslager till vår IoT-lösning, tillhandahåller Microsoft Defender en referensarkitektur för säkerhetsagenter. Dessa ”agenter” är små program som övervakar och samlar in händelser från våra enheter och, tillsammans med SMT, gör det möjligt att konfigurera våran säkerhet från molnet. Vi blir instruerade att koppla upp en C# baserad agent som vi kopplar mot våran IoT-hubb. Här behöver vi de symmetriska nycklar vi skapade i föregående steg. Utöver de säkerhetsrisker som beskrivits ovan med dessa nycklar är det även viktigt att komma ihåg att alla nycklar (samt lösenord) behöver någonstans att förvaras. Enligt labbinstruktionerna ska de förvaras i ett textdokument. Detta gör oss sårbara för obehörig åtkomst och man bör fundera på att använda en alternativ förvaringsplats som till exempel en PEM-fil (Privacy-Enhanced Mail) eller liknande.

Microsoft Defender ger oss möjligheten att skapa så kallade ”Custom Alerts”. Denna funktion är ett tillägg till den redan inbyggda övervakning och säkerhetsanalys som Defender redan ger. Om vi har god förståelse för hur våra IoT-enheter fungerar och beter sig kan detta vara ett mycket användbart sätt för att hitta avvikelser som kan antyda säkerhetsbrister. I labbscenariot sätter vi upp ett ”Custom Alert” för vår temperatursensor eftersom vi redan vet att den förväntade telemetriöverföringsfrekvensen beräknas ligga på mellan 1 till 5 meddelanden över ett 5-minutersintervall. I det här fallet ger det oss ett ytterligare skalskydd mot till exempel överbelastningsattacker (DDoS), som annars är svåra att försvara sig emot, och bidrar också till att hålla nere den mängd data vi lagrar.

Slutligen, i samband med temperaturmätningen i vårt lager, konfigurerar vi en konsolapplikation som skickar telemetri var tionde sekund och testar den regel vi satte upp i ovan stycke. I denna applikation hård-kodar vi in en anslutningssträng (”Primary Connection String”) som används i API-anrop och gör att vår temperatursensor kan kommunicera med IoT-Hubben. Här är det viktigt att känna till att information som lösenord och krypteringsnycklar utgör en mycket stor risk att ha hård-kodade. Det skapar en potentiell ”backdoor” till vår enhet som gör att en angripare kan få access och utföra skadlig aktivitet. För att skydda sig mot detta, kan man lägga nyckeln i en helt annan fil och med hjälp av en variabel i källkoden referera till nyckeln. Detta minskar exponeringen av våra lösenord och exekverar programmet på ett säkrare sätt.

Slutsats

I nästan varje delmoment av labben hanterade vi olika säkerhetsnycklar och lösenord. Det blir uppenbart att hanteringen av dessa utgör en överhängande risk för obehörig åtkomst om det inte sker efter tydliga riktlinjer. Genom att sätta upp en stark företagspolicy kring säkerhet med tillämpning av principer som ”least privilege” (behörighetsanpassad användning för varje användare) och ”least functionality” (endast nödvändig funktionalitet) kan vi reducera vår attackyta och förebygga hotbilden.

Däremot är detta troligtvis lättare sagt än gjort med tanke på den fragmentering som existerar inom IoT. Enhetsresurserna kan snabbt växa till ett omfattande och svåröverblickbart nätverk, som i sin tur leder till att olika delar av lösningen inte hänger med i varandras utveckling. Min uppfattning är att Microsoft Defender kan fungera som ett verktyg för att hjälpa oss en bit på vägen att tackla dessa problem och hålla lösningen inom ett mer väl definierat område. Men, om bristen på kunskap är för stor kan Microsoft Defender inte ge oss detta per automatik. Vi måste själva, som organisation, implementera standarder och utbildning för vår personal i syfte att stärka och främja en kultur där den ”mänskliga faktorn” utgör grunden i det säkerhetsarbete som genomförs.

Något Microsoft Defender gör väldigt bra, är att allt vi kopplar upp mot vår IoT-hubb automatiskt relateras till andra Azure-tjänster kopplat till vår IoT-lösning. Detta kan underlätta säkerhetsintegrationen med andra delar av verksamheten och bidrar till att förbättra vår övervakningsförmåga på en större del av vår IT-infrastruktur. För min del har detta bidragit till en bättre inblick i hur stort användningsområdet för Azure faktiskt är, och att det finns mycket intressant potential att ta del av även i delar som inte nödvändigtvis berör IoT.

En annan viktig reflektion jag har gjort, är den förmåga för testning som blir möjlig med Azure. Med tanke på att IoT-enheter ofta har mycket liten processorkraft och kapacitet, något som kan leda till svårighet i genomförandet av enkla säkerhetsåtgärder, blir Azure tillsammans Microsoft Defender ett bra första steg för att testa sin säkerhet. Precis som vi gjort i labben, kan man enkelt tillämpa ett testcase genom att simulera IoT-enheterna med virtuella maskiner och på så sätt genomföra så kallad ”fuzzing”. Det innebär att man med flit gör fel i användningen av de simulerade enheterna för att se hur säkerheten hanteras och på så sätt upptäcka eventuella brister i sin lösning som man annars kanske inte hade upptäckt innan det varit försent.

Avslutningsvis, har denna laboration framförallt påvisat att konfigurationen av en IoT-lösning innebär fler säkerhetsmoment än jag tidigare trodde, och återspeglar även hur lätt det är att göra misstag eller missa något i sin uppsättningsfas. Även om verktyg som Azure och Microsoft Defender kan hjälpa oss en bra bit på vägen för att säkra vår lösning, är det ändå inte tillräckligt om det råder okunskap kring någon del i flödet. Jag tar i synnerhet med mig en större medvetenhet kring riskerna och betydelsen av att så tidigt som möjligt i utvecklingsfasen tänka på säkerhetsaspekter i min framtida karriär som IoT-utvecklare.

Källförteckning

Muntlig information och föreläsningsmaterial från utbildare Ludwig Simonsson:

https://studentportal.nackademin.se/pluginfile.php/309438/mod_resource/content/1/IT-S%C3%A4kerhet-IoT22.pdf

Labb:

https://microsoftlearning.github.io/AZ-220-Microsoft-Azure-IoT-Developer/Instructions/Labs/LAB_AK_18-azure-defender-for-iot.html#lab-instructions

Externa källor:

https://www.microsoft.com/en-us/security/blog/?p=116871&culture=en-us&country=us

https://learn.microsoft.com/en-us/azure/iot-fundamentals/iot-security-best-practices

https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE50s7O?culture=en-us&country=us

https://arxiv.org/pdf/2202.12501.pdf

https://codeql.github.com/codeql-query-help/csharp/cs-hardcoded-connection-string-credentials/