Analys av Popcorn Time med hjälp av sniffingprogrammet Wireshark

Kevin Å. Kimaryo
Lund University
dat14kak@student.lu.se

Fredrik Siemund
Lund University
htx12fsi@student.lu.se

***Sammanfattning*—Vi har använt sniffingprogrammet Wireshark för att analysera vad som händer när man streamar (svenska: strömmar) film med den olagliga tjänsten Popcorn Time. Popcorn Time är en BitTorrent-klient vilket betyder att användarna både laddar upp och ner data vid tittandet av en film. Popcorn Time hittar andra användare genom att ta kontakt med en server som i BitTorrent-protokollet kallas för "tracker", vilket är det enda centrala elementet i Popcorn Time och som håller reda på vart andra användare finns. Vårt resultat visar att Popcorn Time i princip beter sig som en vanlig BitTorrent-klient med den enda skillnad att tjänsten dessutom använder ett protokoll för streaming av video, kallat GVSP.**

I. Inledning

Att streama innehåll över nätet blir allt vanligare. Tjänster som Netflix och Viaplay har ökat explosionsartat de senaste åren, men även illegala alternativ som Swefilmer, Dreamfilmhd och Popcorn Time har ökat i samma takt och beskrivs numera som ett större problem än den traditionella, illegala fildelningen.[1]

Popcorn Time är ett program där användarna kan titta på film och serier. Utbudet är väldigt stort, användargränsnittet ser profesionellt ut och tjänsten är kostnadsfri; det ser ut och används som en vanlig streamingtjänst. Det som skiljer Popcorn Time från andra streamingtjänster är dock att programmet i grund och botten är en BitTorrent-klient. Detta innebär att användarna inte bara hämtar hem data för att kunna spela upp en film utan att de även aktivt deltar i fildelning genom att själva ladda upp delar av filmen till andra användare. Popcorn Time lanserades 2013 men togs bort igen mars 2014. Efter det har det uppstått en del avknoppningar, utvecklarna till den ursprungliga tjänsten har dock gett sitt stöd till popcorntime.io. I denna undersökning har popcorntime.io använts.

Eftersom Popcorn Time är en BitTorrent-klient sker nästan all kommunikation och dataöverföring peer-to-peer, det vill säga att det inte laddas ner stora datamängder från en central server. Här kan det därför vara relevant att kolla lite närmare på vilken infromation som delas med andra. Detta har vi gjort med hjälp av sniffingprogrammet Wireshark och resultatet presenteras i denna rapport.

II. Teori

Den klassiska bilden av att "ladda ner något från internet" är att en fil skickas från en värd, exempelvis en server eller en annan dator, till ens egna dator. I många fall är det också så det funkar, men är det en stor fil som laddas upp i områden med låga överföringshastigheter kan det innebära stora belastningar på internettrafiken och andra användare i området kan uppleva störningar. Dessutom kan det vara väldigt dyrt att tillgodose den nödvändiga uppladdningshastigheten för att kunna distribuera stora filer på ett snabbt och smidigt sätt.

Istället för att ha en enda nedladdningskälla så kan man utnyttja de användare som laddar ner filen; man låter dem inte bara ladda ner utan även ladda upp delar av filen till andra användare. Detta är tanken bakom BitTorrent och innebär att användarnas (det vill säga de som laddar ner en fil) uppladningskapacitet utnyttjas och på så sätt skickas och hämtas små delar av den stora filen från många användare samtidigt. I BitTorrent-protokollet kallas andra användare som laddar ner för "leechers" och de små delarna som laddas ner kallas för "pieces", alltså bitar. Dessa bitar kan laddas ner och fördelas i slumpmässig ordning. Klienten, det vill säga det BitTorrent-program som används för att ladda ner och upp filer, sätter sedan ihop filen i rätt ordning. Klienten bestämmer i vilken ordning bitarna ska komma och det finns en handfull olika algoritmer för att få en så hög nedladdningshastighet som möjlig. Popcorn Time är en sådan klient och eftersom filen inte bara ska laddas ner utan även spelas upp direkt så måste Popcorn Time, i alla fall till en början, ladda ner bitarna i rätt ordning för att bygga upp en buffert. Detta är en av anledningarna till varför filmen inte börjar direkt till skillnad mot exempelvis Netflix.

A. BitTorrent-protokollet

BitTorrent-protokollet (BT-protokollet) är en essentiell del av Popcorn Time och är det som har gjort tjänsten möjlig. BT-protokollet är ett applikationsprotokoll som sköter kommunikationen mellan två klienter. För att ladda ned en fil via BitTorrent så laddas först den så kallade torrent-filen ned. Detta kan göras på olika sätt, t.ex. via internet eller mail. torrent-filen innehåller information som exempelvis filnamn, storleken på bitarna (vanligtvis 256 kb) och URL-adressen till den så kallade "trackern". Trackern är en server som inte är involverad i filöverföringen men som håller koll på vilka "peers" som finns. En peer är en annan klient som laddar laddar ner och upp samma fil.

Så fort torrent-filen har öppnats av klienten tar den kontakt med trackern och begär en lista med peers. Varje peer har ett Peer ID som generas första gången klienten

startas. Sedan införandet av ”Distributed sloppy hash table” (DHT) för sparande av peer-information har trackers delvis blivit redundanta men används fortfarande för att skynda på processen att hitta peers. Förenklat fungerar DHT genom att varje klient sparar ett litet antal andra klienter - i DHT kallas de för ”nodes” - som befinner sig nära en själv. Varje nod (det vill säga klient) har en routingtabell med närliggande noder och en förfrågan skickas till dem om noden vill hitta peers till en torrent. [2]

Första meddelandet som skickas mellan två peers via BT-protokollet måste vara en så kallad ”handshake”, som bland annat innehåller Peer ID. Efter en handshake kan meddelandet ”Bitfield” skickas som innehåller information om vilka bitar som redan har laddats ner. För att sedan styra vilka bitar som ska skickas och vilka som ska tas emot används meddelanden ”choke”, ”unchoke”, ”interested” och ”not interested”. Utgångsläget efter en handshake är ”choked” och ”not interested”. När en peer har skickat ”choke” innebär det att den inte svarar på några requests som kommer från denna klienten. ”Interested” innebär att en peer är intresserad av bitar som denna klient har att erbjuda och att den kommer att skicka requests så fort klienten har ”unchoked” denna peer. Övriga meddelanden som BT-protokollet kan skicka sammanfattas i tabell 1. [3]

Tabell I
NÅGRA AV DE MEDDELANDEN SOM BT-PROTOKOLLET KAN SKICKA.

Keep alive	Förbindelsen mellan två peers kan stängas om inget skickas på ett tag. För att detta inte ska ske kan Keep alive användas.
Have	Verifikation på att en piece har tagits emot.
Request	Skickas när man vill ha en piece från en peer.
Piece	Används när man skickar en piece.
Cancel	Negativt svar på en request.

BT-protokollet används alltså för att kontrollera och styra flödet av pieces mellan två peers. Inget utbyte av data sker dock i BT-protokollet utan detta sker istället via UDP. En vanlig storelek på en piece är 256 kB (detta kan dock variera och bestäms vid skapandet av torrent-filen) vilket innebär att många UDP-paket behövs för att skicka en piece, något vi även kunde observera i Wireshark.

Ett annat protokoll som dyker upp under analysen är GVSP, vilket står för GigE Vision Streaming Protocol. GVSP är en del av GigE Vision-interfacet som används som ramverk vid dataöverföring av bland annat video. GVSP-protokollet används i det sammanhanget för att definiera datatyper och hur bilder ska skickas. Bilderna skickas sedan via UDP.[6] GVSP används alltså bland annat vid streaming av video.

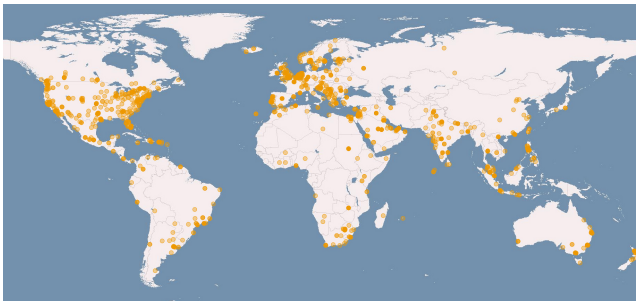
III. METOD

Vi började med att stänga av alla applikationer som använde sig av internet så att Wireshark inte skulle rapportera trafik från dessa. Sedan lät vi Popcorn Time startas upp innan vi startade Wireshark. Detta då vi var intresserade av vad som händer när man börjar streama en film i Popcorn Time. Vi startade sedan Wireshark och satte igång en film i Popcorn Time. Vi hade igång Wireshark i ca 267 sekunder (4 minuter och 27 sekunder) från och med att vi startade filmen. Filmen hann spelas upp i 1 minut och 50 sekunder innan vi avbröt.

Vi använde oss av en vanlig Wi-Fi förbindelse när vi kollade på Popcorn Time.

IV. RESULTAT

På de 267 sekunder som Wireshark stod på så skickades och togs det emot drygt 200 000 paket. Alla paket som skickades använde sig av antingen transportprotokollen UDP (94.2 %) eller TCP (5.8 %). Paketen som använde UDP kom från och skickades till IP-adresser i hela världen (se figur 1), medan de flesta av TCP-paketen kom från servrar i framförallt Kalifornien tillhörande företaget CloudFlare. Majoriteten av alla TCP-paket kom i början av streamingen, innan filmen hade startat.



Figur 1. Figuren visar hur de UDP-paket som skickas kommer och tas emot från hela världen. Bilden är genererad med GeoIP som är ett plug-in till Wireshark.

Under vår analys med Wireshark kunde vi se att trackern, alltså den server som håller koll på alla peers, i vårt fall var en server från företaget CloudFlare. CloudFlare har bland annat hamnat på sjunde plats i världen bland IT-tjänsteföretag för att ha högst koncentration av skadlig och illegal aktivitet samt fått starkt kritik för att hosta tre av ISIS chattforum. [4] [5]

Precis när filmen startades så skickades många paket med applikationsprotokollet HTTP, vilket använder sig av transportprotokollet TCP. Paket nr 440 var ett ”GET .torrent” HTTP-paket (se figur 2). Detta paketet skickades till vår tracker CloudFlare och kan tolkas som en begäran av torrentfilen. Efter detta så började paket som använde sig av transportprotokollet UDP skicaks och tas emot.

Svaret på vår GET-request kommer som paket nr 10 785 och innehåller information om andra peers. Det skickades totalt 25 HTTP-paket innehållande information om andra peers.

```
49293→80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
GET /torrent/B4A81D27B29589DD704A84498780ED183F12EB69.torrent HTTP/1.1
80→49292 [ACK] Seq=1 Ack=373 Win=43648 Len=0 TSval=2894780689 TSecr=59
```

Figur 2. Figuren är ett utklipp ur Wireshark flödet och visar paket 440 som är HTTP-paketet med ”GET torrent” kommandot.

Första paketet som använde sig av applikationsprotokollet BitTorrent (som går via TCP) var paket nr 2912. Detta

var en Handshake. Därmed kom första Handshaken innan vi fått listan på andra peers från vår tracker. Från och med då skickades och togs det emot till största del UDP-paket. Med jämna mellanrum kom det fler BitTorrent paket. Det görs totalt 57 Handshakes under de 2 minuterna som filmen streamades och det krävs en handshake åt båda hållen för att två peer ska kunna utbyta information. I Popcorn Time kunde vi se att vi som mest hade 28 peers. De flesta av våra Handshakes gjordes i ett tidigt stadie av stramingen. Det gjordes inga Handshakes under de sista 50 000 paketen som skickades eller togs emot.

Med regelbundet avstånd dyker det upp paket av typen GVSP, total 7,4 % - 15 000 st - av alla paket som skickas under filmens gång.

Genomsnittliga nedladdningshastigheten var 3.687 Mbit/s (= 461 kB/s) och genomsnittliga uppladdningshastigheten var 263 kbit/s (= 32 kB/s). Nedladdningshastigheten ökade efter hand som vi hade filmen igång. 49.7 % av alla paket som skickades togs emot av oss, resterande 50.3 % skickade vi. Trots att vi skickade fler paket än vad vi tog emot så stod dessa 49,7 % för 93.3 % av den totala datan som skickades/togs emot under filmens gång, det vill säga vi laddade ner mer än vad vi laddade upp.

V. Diskussion

Popcorn Time använder sig av applikationsprotokollet BitTorrent för att kontakta andra användare i världen. Det är sedan dessa användare eller peersen som man laddar ner filmen från. Popcorn Time är alltså peer-to-peer baserat.

I ett tidigt stadie av streamingen skickar Popcorn Time ett GET-torrent HTTP-paket till en IP-adress som ägs av Cloud Flare. Cloud Flare är alltså trackern till den film som vi streamade. De 25 text/plain paketen med de olika peersens IP-adresser kommer också från Cloud Flare. Att det första BitTorrent paketet med Handshake görs innan det första text/plain HTTP-paketet möjliggörs av Distributed sloppy hashtable. Att vi under streamingen gör totalt 57 Handshake tyder på att vi bör ha 28 peers (en måste ha misslyckats då vi inte fick ett Handshake tillbaka). Detta stämmer bra då det stod i Popcorn Time att vi hade just 28 peers.

Det är enkelt att läsa av text/plain paketen då HTTP inte har någon krypteringsfunktion. Ens peers IP-adresser är alltså enkelt att läsa av då man använder sig av Popcorn Time.

Som framgår av resultatet så användes endast transportprotokollen UDP och TCP varav 94.2 % använde UDP. TCP används framförallt i början och väldigt sällan när filmen väl har körts ett tag. Anledningen till detta tror vi är att det i början skickas och tas emot många paket som är essentiella för att peers ska kunna hittas och att filmen ska kunna laddas ner. Det behövs flödeskontroll och felkorrigering för dessa paket vilket TCP erbjuder, det räcker inte med ett best effort transportprotokoll som UDP.

När peers har hittats och viktig information har hämtats såsom undertexter så är det inte lika viktigt med flödeskontroll och felkorrigering. Streamingen måste kunna buffra snabbt och att använda TCP för att hämta varje paket tar helt enkelt för lång tid. Istället används BitTorrent-protokollet för att säga till en peer att man behöver en piece (observera att en piece behöver skickas med många UDP-paket). Detta görs med meddelandet Request i BitTorrent-protokollet. Detta paket, liksom resterande av BitTorrent paketen, är väldigt viktiga paket och behöver därav den säkerhet som TCP erbjuder. Alla de paket som motsvarar en piece skickas sedan med transportprotokollet UDP. En viss flödeskontroll blir det ändå eftersom BitTorrent protokollet skickar sin Have först när den fått en piece.

Ett annat protokoll som dyker upp vid användningen av Popcorn Time är GVSP, GigE Vision Streaming Protocol. Efter mycket efterforskningar så har vi tyvärr inte lyckats ta reda på i vilket syfte Popcorn Time använder sig av GVSP. GVSP är ett protokoll som bland annat används vid streaming av videos. Under den tiden som vi tittade på filmen skickades och togs det emot totalt 8,2 MB GVSP-paket - varav 7,6 MB skickades från vår dator - vilket är relativt lite med tanke på att vi totalt laddade ner 115 MB under tiden som vi tittade på filmen. I nuläget kan vi bara spekulera kring vad protokollet används till. Enligt Wireshark skickas GVSP-paket till många olika privatanvändare i världen. Det finns även en onlineversion av Popcorn Time som fungerar i webbläsaren och vi har funderat ifall det har något med det att göra, men detta är som sagt endast våra egna teorier och inget vi har kunnat visa. Om detta hade varit ett större projekt så hade vi undersökt detta närmare.

Att Popcorn Time är peer-to-peer gör att prestandan väldigt mycket beror på hur många andra peers det finns. Finns det tillräckligt många av dessa så är prestandan väldigt bra och filmen laddas och spelas upp utan avbrott. Finns det inte tillräckligt många av dessa kan det däremot uppstå avbrott då filmen laddar ner en buffert. Något som dessutom påverkas av att Popcorn Time är peer-to-peer är att filmen inte startar omedelbart. I vårt fall fick vi vänta i nästan en minut innan filmen började eftersom den till en början letade efter peers, och då analyserades ändå en relativt populär film.

Kommunikation till trackern sker till största del via HTTP vilket inte är krypterad, det går därför bland annat att läsa att vi efterfrågar en viss torrent. Detta kan vara av intresse för vissa att veta, exempelvis myndigheter eller filmbolag om det är en film användaren inte har betalat för. Kommunikationen mellan peers sker sedan nästan uteslutande med BitTorrent- och UDP-protokollet. I dessa finns det för utomstående inte så mycket intressant information att hitta.

Att nedladdningshastigheten ökade allt eftersom beror på att vi hittade fler peers efterhand, vi fick alltså större nedladdningskapacitet. Nedladdningshastigheten var relativt snabb, detta kan bero på att vi hade igång en populär film där det fanns många peers.

Popcorn Times användargränssnitt kan lätt få en att tro att applikationen är en streamingtjänst. Att Popcorn Time använder sig av BitTorrent-protokollet, och därmed ser till att användarna laddar upp såväl som ner, kan därmed ses som ett oönskat beteende. Att det dessutom är enkelt att utläsa sina peers IP-adresser ur text/plain HTTP-paketet kan också

betraktas som ett oönskat beteende. Anonymitet är antagligen något som många användare av olagliga tjänster som Popcorn Time prioriterar högt.

Om vi skulle fortsatt vår analys av Popcorn Time så hade det varit intressant att analysera paketen som skickas och tas emot under en hel film. Vi hade också velat streama filmer som inte är så populära för att se hur Popcorn Time hade löst streamingen av en film där det inte finns några peers. Det hade även varit spännande att se vad som händer då man startar och stänger Popcorn Time. Fortsätter Popcorn Time att ladda upp från en film som man en gång har kollat på? Slutligen hade vi även velat komma fram till exakt vilken roll GVSP-protokollet spelar då man streamar innehåll via Popcorn Time.

VI. Slutsats

Sammanfattningsvis kan man säga att Popcorn Time i stort sett beter sig som en BitTorrent-klient. En torrent-fil laddas ner, Popcorn Time kontaktar trackern och därefter börjar utbytet av information genom peer-to-peer. Det som skiljer sig är att filen laddas ner från början till slut och att ett streamingprotokoll används vars syfte inte kunde bestämmas i denna undersökning. Detta innebär att man som användare av Popcorn Time deltar i illegal fildelning genom att laddda upp data till andra användare, även om programmets utseende kan få en att tro att det rör sig om en ren streamingtjänst.

Referenser

[1] Joachim Sundell, 2015, *Svenskar ska bötfällas för olaglig streaming*, http://www.svt.se/kultur/svenskar-ska-botfallas-for-olaglig-streaming

[2] Andrew Loewenstern, Arvid Norberg, 2008, *DHT Protocol*, http://www.bittorrent.org/beps/bep_0005.html

[3] Bram Cohen, 2008, *The BitTorrent Protocol Specification*, http://www.bittorrent.org/beps/bep_0003.html

[4] Evan F. Kohlmann, 2015, *Charlie Hebdo and the Jihadi Online Network: Assessing the Role of American Commercial Social Media Platforms*, http://docs.house.gov/meetings/FA/FA18/20150127/102855/HHRG-114-FA18-Wstate-KohlmannE-20150127.pdf

[5] CyberDefcon, 2014, *HostExploit's World Hosts Report*, http://hostexploit.com/downloads/world_hosts_report_201403.pdf

[6] Global association for vision information, *Vision standards, GigE Vision*, http://www.visiononline.org/vision-standards-details.cfm?id=141&type=5