

MICT1

EXERCISE WEEK 5

Joeri van Grimbergen (1244825) | Nursize Bilen (1260235) | Rick van Gorp (1328417)
ZUYD HOGESCHOOL | HEERLEN | GROEP 4

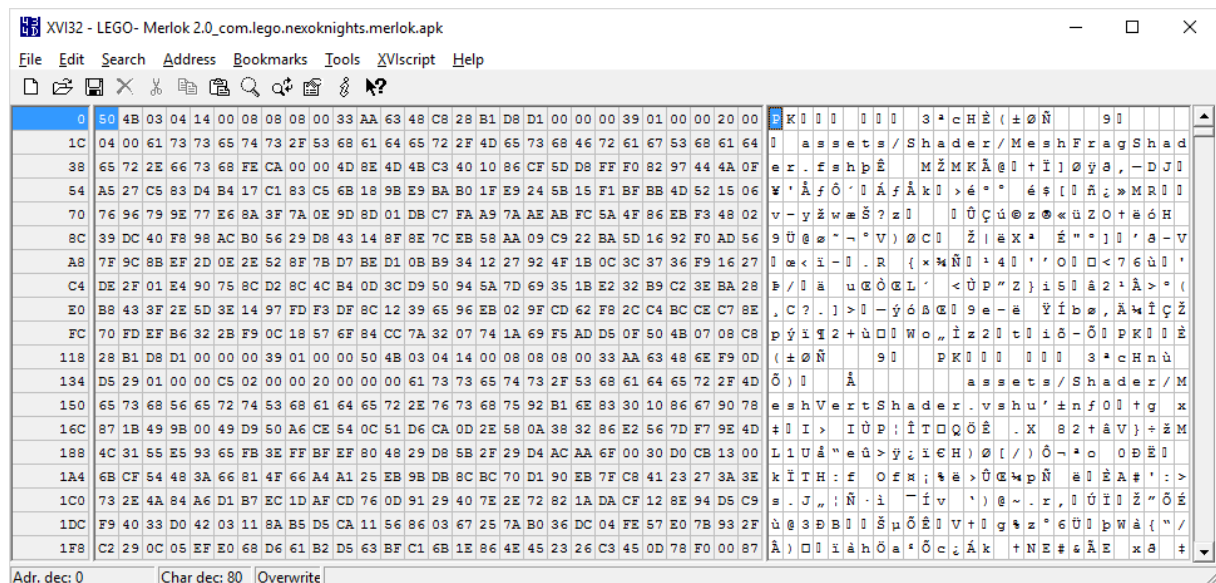
Inhoud

Voorbereiding van analyse	2
Tools	3
Ideeën afhandeling scannen van schilden	4
Shields.dat en Shields.xml	4
Vuforia_Dataset_info & QCAR	5
Shape contour-corners	5
bit-locations	6
bit-radius	6
blockLength	6
messageLength	7
Experimenten	8
Fiddler: Onderscheppen van netwerkverkeer	10
Analyseren van de assetbundle	12
.NETReflector: Decompilen Assembly-Csharp.dll	13
Public Class ShieldScanner	14
Conclusie	23
Bronnen	24








Voorbereiding van analyse

Om de Android applicatie te analyseren is ervoor gekozen om de APK-file te decompilen, zodat de broncode van het spel inzichtelijk kan worden gemaakt. Om het APK-bestand op te halen van het toestel is gebruik gemaakt van de app “APKExtractor”. Deze applicatie is in staat om van reeds geïnstalleerde apps de APK-bestanden op te halen en op te slaan op een aangegeven locatie.

Een APK-bestand is qua structuur gelijkwaardig aan een ZIP-bestand. Dit is te identificeren aan het gebruik van een PK header en dezelfde opbouw zoals gedefinieerd in de PK file specificatie van PKWare.



Het uitpakken van het APK-bestand met behulp van WinRAR, leverde de volgende resultaten op:

	assets	14-3-2016 15:54	Bestandsmap	
	lib	14-3-2016 15:54	Bestandsmap	
	META-INF	14-3-2016 15:54	Bestandsmap	
	res	14-3-2016 15:54	Bestandsmap	
	AndroidManifest.xml	3-3-2016 21:17	XML-document	9 kB
	classes.dex	3-3-2016 21:11	DEX-bestand	5.295 kB
	resources.arsc	3-3-2016 21:11	ARSC-bestand	1.947 kB

De interessante bestanden die geanalyseerd kunnen worden zijn *classes.dex* en de bestanden die staan in de map *assets*. Het bestand *classes.dex* bevat alle broncode dat gerelateerd is aan het APK-bestand. Deze broncode is normaliter in de programmeertaal Java geschreven. De map *assets/bin/Data/* bevat unity gerelateerde bestanden. Dit houdt in dat in de map *assets/bin/Data/Managed/* DLL-bestanden te vinden zijn die representatief zijn voor de broncode van het spel. Met name het DLL-bestand *Assembly-Csharp.dll* is interessant, omdat dit bestand het grootste deel van de broncode van het spel bevat.

Naast deze voorbereiding is gecontroleerd welke bestanden het APK-bestand als uitvoer genereert in de lokale mappen van Android. In de map *Android/Data/com.lego.nexoknights.merlok/Files/QCAR* zijn potentieel interessante bestanden aangetroffen, namelijk: *shields.dat* en *shields.xml*. Deze bestanden kunnen mogelijk meer informatie bevatten over de schilden.

Tools

De tools die zijn gebruikt voor dit onderzoek zijn onderstaand weergegeven:

- XVI32: <http://www.handshake.de/user/chmaas/delphi/download/xvi32.zip>;
- WinRAR: <http://rarlab.com/rar/winrar-x64-520.exe>;
- AssetBundleExtractor (for Unity):
https://mega.nz/#lvQoBkApZ!Ar_vlrN3TfWhz4rMxvE4Ws3pC_5iJDgJeWkCUGp_ODU;
- .NETReflector: <http://www.red-gate.com/products/dotnet-development/reflector/>;
- TextCrawler: http://www.digitalvolcano.co.uk/download/TextCrawler_Setup.exe;
- Dex2Jar: <https://github.com/pxb1988/dex2jar>.

Ideeën afhandeling scannen van schilden

Naar aanleiding van de constatering dat zijn gemaakt tijdens de voorbereiding van de analyse van de broncode achter het verwerken van de schilden zijn enkele ideeën gevormd.

Shields.dat en Shields.xml

In het bestand Shields.dat bevond zich het bestand config.info. Het bestand Shields.dat is uitgepakt met het programma WinRAR. In dit bestand wordt vermoedelijk de configuratie beschreven die gebruikt wordt door de applicatie om de schilden te scannen.

Onderstaand is het volledige script (in XML-formaat) weergegeven:

```
<?xml version="1.0" encoding="UTF-8"?>
<QCARInfo version="1.5" xsi:noNamespaceSchemaLocation="Vuforia_Dataset_info_4.3.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <TargetSet version="4.3">
        <ContourMarker targetId="9dae5ac0df7311e488300800200c9a66" name="shields"
toolVersion="4.3.2" version="4.3" size="31.182 46.135" >
            <Shape contour-corners="M -15.591,15.272 -15.589,-19.574 15.589,-19.574
15.591,15.272 0,30.863" bit-locations="M -12.373,13.748 -12.373,10.998 -12.373,8.249 -12.373,5.499 -
12.373,2.75 -12.373,0 -12.373,-2.75 -12.373,-5.499 -12.373,-8.249 -12.373,-10.998 -12.373,-13.748 -12.373,-
16.498 -9.624,-16.498 -6.874,-16.498 -4.124,-16.498 -1.375,-16.498 1.375,-16.498 4.124,-16.498 6.874,-16.498
9.624,-16.498 12.373,-16.498 12.373,-13.748 12.373,-10.998 12.373,-8.249 12.373,-5.499 12.373,-2.75
12.373,0 12.373,2.75 12.373,5.499 12.373,8.249 12.373,10.998 12.373,13.748" bit-radius="0.97" />
            <Code format="eh" blockLength="32" messageLength="26" minDistance="4"
alphabetSize="2">
                <Restrictions>
                    <![CDATA[0x0167e6b2 0x02881172 0x02381c72 0x0203c072
0x038bd1f2 0x033bdcf2 0x030000f2 0x01981992 0x02fc3f52 0x02c7e352 0x0277ee52 0x024c3252 0x03ffffd2
0x03c423d2 0x000bd022 0x01b81da2 0x0183c1a2 0x0133cca2 0x010810a2 0x02e7e762 0x01c4239e
0x02435bee]]>
                </Restrictions>
            </Code>
        </ContourMarker>
    </TargetSet>
</QCARInfo>
```

Een aantal dingen zijn ons hier opgevallen, deze zijn voor het gemak in het geel gemarkeerd.

Vuforia_Dataset_info & QCAR

In het XML-bestand werd gerefereerd naar een Vuforia Dataset. De zoekopdracht “Vuforia” op Google leidde naar een resultaat dat betrekking had op een Tracking systeem dat in staat is om met behulp van een camera informatie af te lezen uit herkenbare onderdelen uit een omgeving. Dit is een algoritme dat bijvoorbeeld gebruikt kan worden in Unity.

De structuur van de schilden, zoals onderstaande weergegeven, komt erg overeen met de beschrijving van “Frame Markers” op de website van Vuforia (<https://developer.vuforia.com/library/articles/Training/Frame-Markers-Guide>).



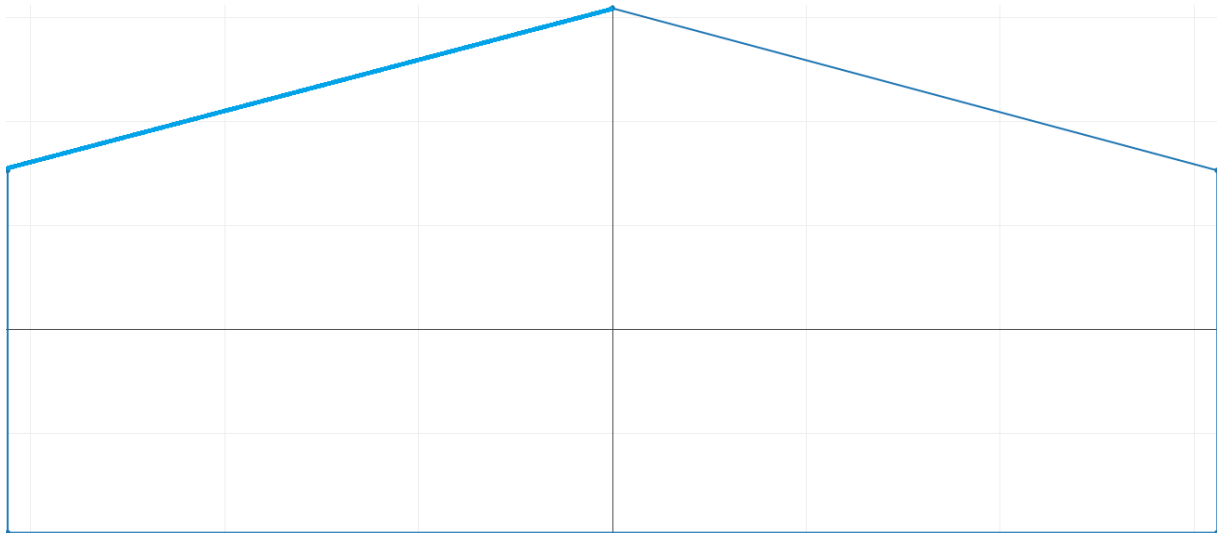
Deze frame markers kunnen ook gescand worden met behulp van een camera op basis van vooraf gedefinieerde instellingen. De instellingen die zijn weergegeven in het bestand *config.info* kunnen dus gerelateerd zijn aan de frame marker instellingen van een schild. Dit is getest op de volgende wijze:

- Op de Android telefoon zijn de bestanden *Shields.dat* en *Shields.xml* verwijderd van de eerder aangegeven locatie (*Android/Data/com.lego.nexoknights.merlok/Files/QCAR*);
- De gedefinieerde *size* bij element *ContourMarker* is gewijzigd van: 31.182 46.135 naar 20.000 60.000;
- Het bestand *config.info* is terug in het bestand *Shields.dat* gestopt;
- Beide bestanden zijn teruggeplaatst op de locatie waar de bestanden verwijderd zijn (*Android/Data/com.lego.nexoknights.merlok/Files/QCAR*);
- De Lego applicatie is geopend, waarbij is getest of de schilden nog te scannen waren. Dit was niet het geval.

Uit bovenstaand onderzoek kan geconcludeerd worden dat het bestand *config.info* instellingen bevat die gerelateerd zijn aan het scannen van de schilden in de vorm van een frame marker. Er wordt gebruik gemaakt van het Vuforia algoritme door de applicatie om de schilden te scannen.

Shape contour-corners

Wij vermoeden dat dit de 5 hoeken van het schild aangeven in coördinaten, als we deze shape na proberen te maken in een teken tool dan komt er ongeveer deze shape uit:



Er is geen verklaring waarom dit schild op zijn kop is afgebeeld, maar het lijkt er wel op dat deze coördinaten de hoeken van het schild aangeven. De reden dat de vorm van het schild groter uitvalt dan het formaat van het schild daadwerkelijk is, is doordat er eerder in de *config.info* binnen het element *ContourMarker* een grootte van het schild wordt gedefinieerd. Deze grootte toegepast op bovenstaande afbeelding resulteert in een schild van hetzelfde formaat als de originele schilden.

bit-locations

Dit zijn wederom coördinaten, volgens ons zijn dit de locaties aan de rand van het schild waar de data zou moeten staan die gescand wordt, als we deze coördinaten tellen dan komen we ook op 32 locaties uit.



In bovenstaande afbeelding is een voorbeeld van een schild weergegeven. Wat bij het bekijken van dit schild opvalt is dat er ruimte is voor 32 vierkantjes. Dit komt overeen met het aantal bit-locations en de gedefinieerde *blockLength*.

bit-radius

Hier wordt de size aangegeven van de bit locaties, deze heeft een waarde van 0.97.

blockLength

De *blocklength* heeft als waarde 32, wat overeenkomt met de bit-locations. Aangezien de blocks gevuld of leeg kunnen zijn (0 of 1) kunnen we concluderen dat er $2^{32} = 4.294.967.296$ mogelijke combinaties zijn.

messageLength

De messagelength heeft als waarde 26, deze kunnen wij niet volledig verklaren. Wel zijn we na onderzoek er achter dat de schilden altijd met de binaire waarde 00010 beginnen. Onderstaand zijn de binaire waarden van 23 willekeurig gekozen schilden:

```
00010010101100111001011101101100
00010011101100110010001001110100
00010111010010011010101001100101
00010100010110010100110100010110
00010010100100110010110100110110
00010010100111000101010001000110
00010100100101001010110110101010
00010010010110110001100010101011
00010010001001001101001010100101
00010001010100110101010110111000
00010110011001010011100110010100
00010001101001010110101001110111
00010010110010010101110111010011
00010100110010101010011101001100
00010001011010100010011010101000
00010101011101110110110100011101
00010100110101100101110110111011
00010001110010010001010001101100
00010011101110100110011100111011
00010100111001100111010110101001
00010010110100011011001010001110
00010010100100110010110100110110
00010001101010010101100100101011
```

Als deze schilden met 6 vaste waarden waren begonnen konden we hier concluderen dat messagelength van 26 een logische waarde was, echter kunnen we dit nu niet vast stellen. Wel kunnen we hier mee concluderen dat de mogelijke combinaties van 2^{32} naar $2^{27} = 134.217.728$ gaat. Daarnaast valt op dat de sets met het getal 1 of 0 achter elkaar niet groter zijn dan 3. Dit zou kunnen inhouden dat een zelfde waarde maximaal drie keer achter elkaar mag staan. Dit resulteert

in nog minder mogelijkheden en maakt het enigszins eenvoudiger om de waardes van de schilden door middel van een brute-force aanval te achterhalen.

Experimenten

We zijn er achter gekomen dat de schilden een binaire waarde heeft met een lengte van 32, hieronder geven we een illustratie hoe de schilden afgelezen moeten worden:



We beginnen linksonder te lezen en gaan vervolgens het schild rond, een lege pixel geeft een 0 aan en een gevulde pixel een 1. Dus in dit geval krijgen we de binaire waarde: 0001 0010 0110 0101 1010 1010 0010 0100.

Na onderzoek hebben we een website gevonden waar we deze waarde in kunnen voeren en genereren, hier kunnen we de binaire code invullen en vervolgens het zelfde schild genereren: <http://zombievision.net/nexoknights/> het invoeren dient wel zonder spaties te gebeuren.

Het resultaat is:

← → ↺ 🏠

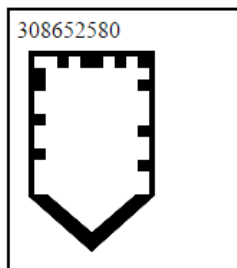
Pick one:

Select a range: Start: End: (Use this with caution!)

- OR -

- OR -

- OR -



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Welke weer te scannen is met de lego nexo knights app op je telefoon. Het nadeel van deze generator is echter dat deze de uitzonderingen, zoals beschreven onder *messageLength* niet verwerkt. Deze generator heeft dus 4.294.967.296 mogelijke combinaties van schilden.

Fiddler: Onderscheppen van netwerkverkeer

Een mogelijke hypothese is dat de applicatie gebruik maakt van een legoserver om de gescande schilden te valideren:

- Optie 1: Schilden worden bij opstarten van het spel opgehaald van een legoserver;
- Optie 2: Schilden worden bij scannen gecontroleerd door informatie op te sturen naar een legoserver;
- Optie 3: Schilden staan vastgelegd in een database in de code of een bestand, lokaal opgeslagen.

Om te controleren of bestanden worden opgehaald bij het opstarten van het spel is een proxy opgezet, waarbij het Android toestel verbinding maakt met de server die in Fiddler draait. Fiddler neemt vervolgens de verbinding over en vervolgd deze opzet. Op het Android toestel is het apparaat geconfigureerd om verbinding te maken met het lokale IP-adres waarop Fiddler draait: 192.168.178.13. Fiddler heeft in de configuratie opgenomen dat het luistert op poort 8080. Daarom dient de proxy server op het Android toestel ook ingesteld te worden op poort 8080. Vervolgens wordt de applicatie (voor de eerste keer) opgestart en zijn onderstaande resultaten gegenereerd:

	94	302	HTTP	lego.com	/nl-nl/gco/203/android/1?format=json	216	private	text/html; c...	[#93]
	95	302	HTTP	lego.com	/en-us/default.aspx?404%3bhttp%3a%2f%2flego.com%3a80%2f/nl-nl%2fgco%2f203%2fandroid%2e...	243	private	text/html; c...	[#94]
	96	200	HTTP	www.gelo.com	/nl-nl/application/config/api/conf/g/hexloknights/app/android/1?format=json&domainredir=lego...	866		application/...	[#95]

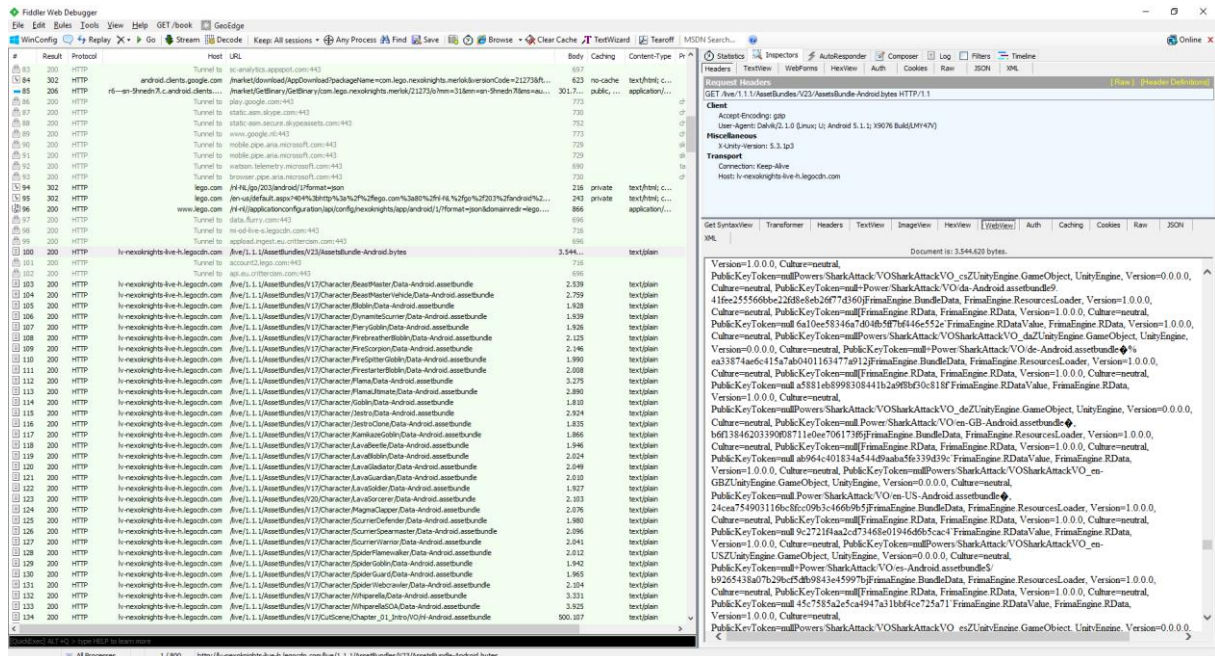
Bovenstaand is de eerste verbinding weergegeven met een legoserver. Tijdens deze verbinding wordt een JSON-bestand opgehaald met de volgende inhoud:

```

{
  "configuration": {
    "@contentlanguage": "nl-NL",
    "@country": "NL",
    "@name": "Nexo Knights",
    "@version": 1
  },
  "commonconfigurations": {
    "appdialogue": {
      "customvalues": {
        "killswitchtext": "Deze app wordt niet langer door de LEGO Groep onderhouden. Gelieve het van uw apparaat te wissen.",
        "minimumversiontext": "Er zijn nieuwe elementen toegevoegd aan deze creatie. Verbeter je app door middel van je winkel",
        "ok": "OK",
        "warning": "Waarschuwing"
      },
      "legoid": {
        "endpoints": {
          "legoidconfiguration": "https://mi-od-live-s.legocdn.com/fr/account2/nl-nl/services/account/apiconfigservice.ashx",
          "legoiddeletedaccountsfeed": "https://account2.lego.com/account/deletedusers/feed",
          "legoidmyfriends": "https://wwwsecure.us.lego.com/nl-nl/myfriends",
          "legoidmyfriendswebpage": "https://wwwsecure.us.lego.com/nl-nl/myfriends?context=app",
          "legoidtoken": "https://account2.lego.com/account/v1/partnertoken"
        },
        "privacypolicy": {
          "endpoints": {
            "cookiepolicy": "http://lego.com/nl-nl/go/7/cookie",
            "privacypolicy": "http://lego.com/nl-nl/go/7/privacy"
          }
        }
      }
    },
    "customvalues": {
      "gameassetshost": "http://tv-nexoknights-live-h.legocdn.com",
      "gameassetshost_cn": "http://aka-cn-cache.lego.cn/e/nexoknightsns",
      "previewfolder": "preview"
    },
    "endpoints": {
      "newsfeedendpoint": "http://www.lego.com/nexoknights/v1/api/articles.json?locale=nl-nl&theme=nexoknights&tag=App&channel=",
      "videofeedendpoint": "http://www.lego.com/nexoknights/v1/api/videos.json?locale=nl-nl&theme=nexoknights&tag=app&channel=Apps",
      "experience": "nexoknights",
      "killswitch": 0,
      "minimumversion": 1.0.0,
      "tracking": 0
    }
  }
}

```

Er worden algemene opties opgehaald waarvan de Lego applicatie gebruik kan maken tijdens het verbinden met de Lego servers. Vermoedelijk verbindt de applicatie later met één van de hierboven gedefinieerde servers om de rest van de data op te halen. De eerstvolgende verbinding die de applicatie opzet is inderdaad met één van de URLs die is beschreven in het JSON-bestand, namelijk: *lv-nexoknights-live-h.legocdn.com*. Daarbij wordt het bestand *AssetsBundle-Android.bytes* opgehaald.



In bovenstaande afbeelding is een deel van dit bestand weergegeven. Het bestand lijkt referenties naar vervolgbestanden die gedownload moeten worden of referenties naar game objecten die reeds zijn toegevoegd aan het spel weer te geven. Dit is vooralsnog onbekend. In bovenstaande afbeelding is wel te zien dat er verschillende bestanden worden gedownload waaronder ook assetbundles die betrekking hebben op powers. De namen van deze powers komen overeen met de namen van de schilden. Voorbeeld van een URL waarvan Power-gerelateerde data wordt opgehaald:

<http://lv-nexoknights-live-h.legocdn.com/live/1.1.1/AssetBundles/V17/Power/SwiftSting/VO/nl-BE-Android.assetbundle>

Naar aanleiding van deze constatering zou gecontroleerd kunnen worden wat er precies in deze assetbundle staat. Een volgende hypothese zou kunnen zijn:

- Deze power-gerelateerde assetbundles bevatten informatie die gebruikt wordt om het schild te identificeren na het scannen ervan.

Optie 1 zou dus een valide mogelijkheid kunnen zijn. Optie 2 en Optie 3 zijn echter nog niet getest op dit moment. Optie 2 wordt getest door in de Lego applicatie de camera functie te openen en daadwerkelijk een schild te scannen. Uit deze test bleek dat er geen internetverkeer plaatsvond ten tijde van het scannen, vlak vooraf het scannen of net na het scannen van het schild. Daarmee kan dus geconcludeerd worden dat Optie 2 niet valide is en dat Optie 1 of Optie 3 plausibel zijn.

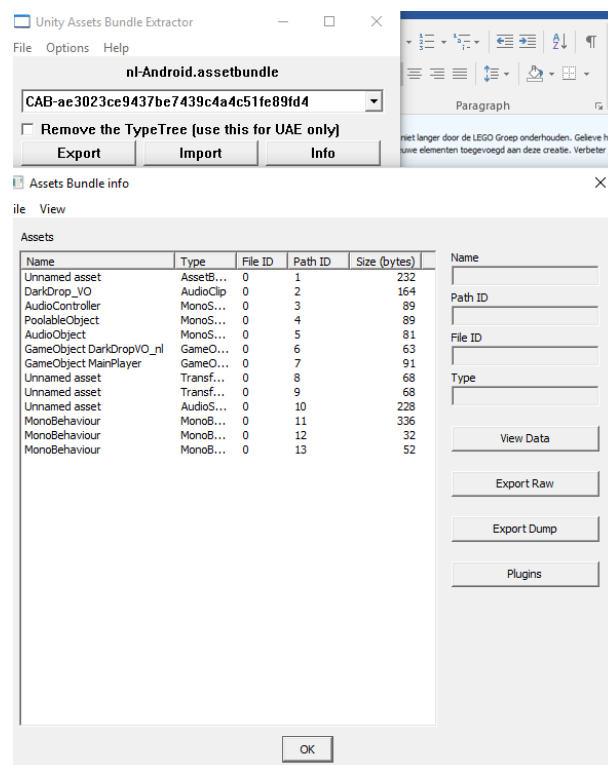
Optie 1 en Optie 3 hebben enigszins overlap, omdat als de schilden worden opgehaald deze schilden ook staan vastgelegd in een bestand of zijn geladen in de code.

Analyseren van de assetbundle

Tijdens het onderscheppen van het netwerkverkeer is geconstateerd dat er bestanden gedownload worden van de legoservers die gerelateerd zouden kunnen zijn aan powers (schilden). Deze bestanden zouden mogelijk data kunnen bevatten die de identificatie van deze schilden binnen het spel mogelijk maakt. De hypothese die hierbij gesteld wordt is:

- Deze power-gerelateerde assetbundles bevatten informatie die gebruikt wordt om het schild te identificeren na het scannen ervan;

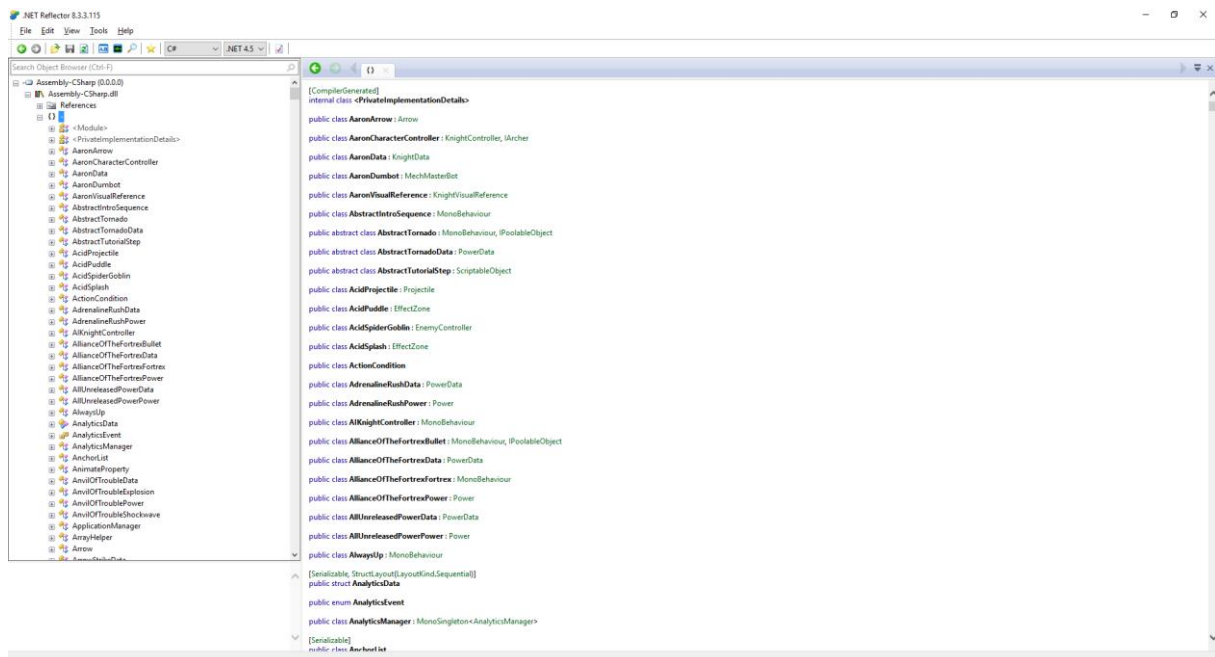
Een eigenschap van deze assetbundles is echter, is dat het een eindproduct is van een gecompileerd spel. Om de data uit deze assetbundles te interpreteren is gebruik gemaakt van de tool *AssetBundleExtractor*. Als één van de assetbundles geopend wordt blijkt dat deze assetbundle gecompressed is. *AssetBundleExtractor* is in staat om deze assetbundle zelfstandig te decomprimeren. De structuur van een assetbundle is in te zien door te klikken op "Info".



Uit de structuur die gebruikt is in deze assetbundle is op te maken dat deze bundel het schildobject voor de game zelf bevat en de bijbehorende geluidsinstellingen en geluiden. Uit deze informatie is niet direct op te maken of het een asset bevat die gebruikt wordt ter identificatie van een schild. De hypothese blijft op dit moment dus onbeantwoord.

.NETReflector: Decompilen Assembly-Csharp.dll

In dit gedeelte wordt beschreven hoe de unity gerelateerde DLL is gedecompileerd en geanalyseerd.



In bovenstaande afbeelding is het bestand, geopend in .NETReflector, weergegeven. Aangezien optie 3: “Schilden staan vastgelegd in een database in de code of een bestand, lokaal opgeslagen” nog openstaat wordt gezocht naar enums binnen de code. Enums zijn lijsten met informatie die snel toegankelijk zijn voor de rest van de broncode. De Enum die gerelateerd is aan schilden en powers is de volgende:

- Epower: Deze enum bevat de namen en locaties van schilden die op dit moment bekend zijn;

```
public enum EPower
{
    NONE,
    THUNDERSHOCK,
    FIST SMASH,
    FIRE TORNADO,
    REFLECTIVE_SHIELD,
    HASTE,
    SEA DRAGON,
    MIGHT_OF_THE_MAGICIAN,
    ULTRA_ARMOUR_AWESOME_SAUCE,
    FLIGHT_OF_THE_PHOENIX,
    CLONING,
    DAZZLING_HERO,
    ROARING_RIGHTEUSNESS,
    AVENGING_ULTRA_ARMOR,
    MAJESTIC_BENEVOLENCE,
    ROLLING_FIREBALL,
    BLADE_OF_BRAVERY,
    DRAGON_OF_JUSTICE,
    PIRANHA_BITE,
    HAMMER_TIME,
    SWORD_TORNADO,
    GOBLIN_ATTACK,
    SLIME_BLAST,
    ARROW_STRIKE,
    TRACTOR_BEAM,
    NEXO_BLADE,
    BACKLASH_LIGHTNING,
    GROUND_POUND,
    BANANA_BOMB,
    TARGET_BLAZER,
    PHOENIX_BLAZE,
    JUNGLE_DRAGON,
    FORCE_FIELD,
    STRONGHOLD_OF_RESOLUTION,
    RAGING_RALLY,
    MAGNETIZE,
    STONE_STUN,
    STORM_DRAGON,
    DYNAMIGHTY,
    INCINERATE,
    SPIRIT_VORTEX,
    FOUL_STEAM,
    GIANT_GROWTH,
    CLAPPER_CLAW,
    HAWK_HOLLER,
    MACE_RAIN,
    LAVA_DRAGON,
    TAKE_OFF,
    RUSHING_STRIKE,
    ICE_DRAGON,
    BACK_FIRE,
    TIME_BREACH,
    VENOM_BITE,
    EGG_OF_DOOM,
    ROCK_RIPPER,
    SWIFT_STING,
    ALLIANCE_OF_THE_FORTREX,
    STARFALL,
    POWER_OF_UNITED_KNIGHTS,
    LION_OF_BRAVERY,
    SHIELD_OF_SCHOOLING,
    ADRENALINE_RUSH,
    GAUNTLET_OF_TRUTH,
    BOMB_BLAST,
    MOTH_SWARM,
    UNUSED_0,
    MECH_MASTER,
    CHICKEN_POWER,
    CHARGING_ATTACK,
    MIGHTINESS,
    FLAME_WRECK,
    BULLFROG_SUPERBOUND,
    BOOMERANG,
    TOXIC_STING,
    END_OF_TIME,
    MINIFY_SURPRISE,
    DOWNSIZER,
    ROCKETSHIP,
    MIGHTY_PEN,
    BEAM_JUMP,
    GREATEST_HITS,
    WHIRLFLASH,
    DARK_DROP,
    VEIL_OF_CONCEALMENT,
    CLOVER_OF_MISFORTUNE,
    BONKERS_BEANS,
    DISCO_FRENZY,
    SABER_SLASH,
    SILVER_WHIP,
    WILD_BOAR,
    PRISM_OF_CLARITY,
    WASP_MISSILE,
    UNDER_WOE,
    CROC_TEAR,
    HYPER_KICK,
    POISON_BURST,
    POWER_PLANT,
    SIDEKICK,
    ICE_RAIN,
    FORMATION_OF_FORTITUDE,
    SOARING_EAGLE,
    UNUSED_1,
    GIGGLING_ULTRA_ARMOR,
    SHIP_WRECKER,
    CHAMPION_OF_CHIVALRY,
    MANOWAR,
    ANVIL_OF_TROUBLE,
    SHADOW_SNAIL,
    SPIN_DRIFTER,
    DAZZLING_LURE,
    STONE_BURST,
    ICE_BURST,
    MAGMA_BURST,
    LIGHT_BURST,
    ULTRA_ARMOR_ACTIVATE,
    MAGIC_ANTI_EVIL_ULTRA_ARMOR_UPGRADE,
    SAND_TORNADO,
    UNUSED_2,
    SPARROW_TORNADO,
    GRIFFON_OF_GRACIOUSNESS,
    CHIMERA_OF_COURTESY,
    BEETLE_BOMB,
    SILK_SPIDER,
    FIREFLIES,
    ROCK_N_ROLL,
    SKUNK_STENCH,
    SOUR_STRIKE,
    GLORY_OF_KNIGHTON,
    SWORD_OF_STRENGTH_AND_SUPREMACY,
    CYCLONIC_STRIKE,
    ZAP_ZAP,
    BAD_BOILS,
    BROOM_OF_DOOM,
    ORDER_OF_THE_KNIGHTS_CODE,
    FLASH_CANNON,
    ALL_UNRELEASED_POWER,
    RAPTOR_BITE,
    GAZE_OF_THE_GORGON,
    ROYAL_BRAWL,
    ATOMIC_ACORN,
    PINBALL_MAGICIAN,
    GOO_GEYSER,
    MOUSE_TRAP,
    BRAINFREEZE,
    BROCCOLI_TORNADO,
    IRON_HAIR,
    SHARK_ATTACK,
    GLOBE_OF_LIGHT,
    WATER_WALL,
    WALL_BLOCK,
    QUAKE_BALL,
    DRAINING_SCARF,
    STONE_SPIKE,
    GOLDEN_TOUCH,
    WHIRLWIND,
    STAMPEDE,
    ROCK_THROW,
    MONSOON_STORM,
    MAX_POWER,
    POTENT_PEPPEY_POWER_AXE,
    COOL_CREATION,
    HAIL_TO_THE_JESTER,
    DARING_DELIVERANCE,
    REMOTE_CONTROL,
    SUPER_HERO_BODYSLAM,
    SERPENT_OF_ANTI_VIRUS,
    SNAKE_DEN,
    FORCE_OF_NATURE,
    BUBBLE_GUM_MISFIRE,
    RAT_TIDE,
    FUNKY_FUNGUS,
    BEAR_CLAWS,
    QUICKSAND,
    SNAPPER_STAND,
    RIPPING_THORNS,
    DEPTH_CHARGE
}
```

Naast deze enum zijn er nog enkele enums te vinden die informatie bevatten over powers, maar die zijn niet relevant voor de identificatie van het schild.

Public Class ShieldScanner

Een belangrijke class die gebruikt wordt bij het proces om schilden te scannen met behulp van de scanfunctie binnen de Lego applicatie is *ShieldScanner*. De mogelijkheid bestaat dat uit deze class is af te leiden op wat voor wijze de invoer wordt geverifieerd.

Onderstaande functie code zou relevant kunnen zijn om te kunnen bepalen hoe wordt bepaald of de gescande code valide is en te koppelen is aan een schild.

```
private bool ValidateIfPowerIsAvailable(ShieldTrackerFeedback aVisualTracker)
{
    uint markerID = aVisualTracker.GetTrackedShieldData().ContourMarker.MarkerID;
    this.m_CurrentScannedPower = PowerUtil.GetByShieldBorderID(markerID.ToString());
    this.m_CurrentScannedPowerLevel = 0;
    bool flag = this.m_CurrentScannedPower != null;
    if (flag)
    {
        MonoSingleton<CritticismManager>.Instance.LoggingBreadcrumbs(MonoSingleton<CritticismManager>.Instance.m_ScanPowerTransaction + " Shield border id found");
        this.EnterWaitingForScanResultState();
    }
    else
    {
        MonoSingleton<CritticismManager>.Instance.LoggingBreadcrumbs(MonoSingleton<CritticismManager>.Instance.m_ScanPowerTransaction + " Shield border id not found: " + markerID.ToString());
        this.EnterShieldNotAvailableState(aVisualTracker, markerID);
        MonoSingleton<CritticismManager>.Instance.FailTransaction(MonoSingleton<CritticismManager>.Instance.m_ScanPowerTransaction);
    }
    aVisualTracker.ClearTrackedShield();
    return flag;
}
```

Het gescande Shield ID (*markerID*) is een unsigned integer. Dit houdt in dat het een getal is en dat er geen letters in zitten. Volgens de beschrijving van Frame Markers correspondeert de waarde van MarkerID aan de binaire waarde die is vastgesteld in het schild. Deze binaire waarde wordt geconverteerd naar een *unsigned integer*. Voorbeeld:



Binaire notatie: 00010011101011010101000101010111
Decimale Notatie (MarkerID): 330125655

Vervolgens wordt na het vaststellen van de MarkerID via Class *PowerUtil* het schild opgehaald aan de hand van het schild ID. Dit wordt uitgevoerd door een call uit te voeren naar functie *PowerUtil.GetByShieldBorderID(markerID.ToString())*. Deze functie is onderstaand weergegeven:

```
public static PowerData GetByShieldBorderID(string aID)
{
    if (mPowerDataByShieldBorderID.ContainsKey(aID))
    {
        return mPowerDataByShieldBorderID[aID];
    }
    return null;
}
```


In deze functie wordt gecontroleerd of variabele *mPowerDataByShieldBorderID* de markerID bevat. De controledata staat dus in variabele *mPowerDataByShieldPowerID*.

De definitie van *mPowerDataByShieldBorderID* in Class *PowerUtil*:

```
private static Dictionary<string, PowerData> mPowerDataByShieldBorderID;
```

Deze variabele bevat dus alle MarkerID's met bijbehorende PowerData. Deze dictionary moet echter ergens gevuld worden. Dit vullen wordt uitgevoerd in functies *AddAdditionalBorderId* en *AddPower*.

```
public static void AddAdditionalBorderId(PowerData aPowerData)
{
    if (aPowerData.AdditionnalShieldBorderID != null)
    {
        foreach (string str in aPowerData.AdditionnalShieldBorderID)
        {
            mPowerDataByShieldBorderID[str] = aPowerData;
        }
    }
}

public static void AddPower(EPower aPower, PowerData aPowerData)
{
    mAllPower.Add(aPowerData);
    if (aPowerData.ShieldBorderID != string.Empty)
    {
        mPowerDataByShieldBorderID[aPowerData.ShieldBorderID] = aPowerData;
        AddAdditionalBorderId(aPowerData);
    }
    else if (aPower == EPower.ALL_UNRELEASED_POWER)
    {
        AddAdditionalBorderId(aPowerData);
    }
    UpdateBonusLevel(aPowerData.PowerColor);
}
```

De dictionary wordt gevuld aan de hand van input PowerData. Om te bekijken waar deze ShieldBorderID uit PowerData vandaan komt, wordt de PowerData Class geanalyseerd. In de PowerData Class is PowerData gedefinieerd als een ScriptableObject. Dit houdt in dat er in een dergelijke Class grote hoeveelheden aan data kunnen worden opgeslagen die gebruikt kunnen worden door meer onderdelen van de code. PowerData is dus te vergelijken met een "object" die veel data bevat. Zoals in de afbeeldingen hierboven weergegeven is bijvoorbeeld variabele *aPowerData* een PowerData object. Één van de componenten van het PowerData object is een ShieldBorderID. Deze wordt gelezen uit het PowerData object dat als argument bij de functie call *AddPower* is meegestuurd.

Deze PowerData kan worden opgehaald met de functie *GetPowerData* die gedefinieerd is in class *EPowermap*. Deze class is een extensie op class EnumMapper. Deze class mapt op basis van de waardes in enum *EPower* (bovenstaand weergegeven) naar een RData object. Op dit punt wordt gecommuniceerd vanuit het script met een GameObject en wordt hier data uit opgehaald. De scripts die in dit stukje worden beschreven zijn weergegeven op de volgende pagina:

Uit class *PowerUtil*:

```
public static PowerData GetPowerData(EPower aPower)
{
    PowerData powerData = EPowerMap.GetPowerData(aPower);
    if (powerData == null)
    {
        DebugNex.LogError(aPower.ToString() + " Not in EPowerMap");
    }
    return powerData;
}
```

Dit stuk code refereert naar de functie *GetPowerData* in class *EPowerMap*. De gebruikte zoekstring is de waarde in de enum *EPower*.

Uit class *EpowerMap*:

```
public static PowerData GetPowerData(EPower aKey)
{
    RData aData = GetValue(aKey);
    if (RData.IsNullOrEmpty(aData))
    {
        Debug.LogError("EPowerMap::GetPowerData. aKey: " + aKey.ToString());
        return null;
    }
    return (aData.target as PowerData);
}
```

In dit stuk code wordt aan de hand van een waarde uit de enum *EPower* een *RData* object gemaakt. De waarde die *aData* gaat bevatten wordt opgehaald door de functie call *GetValue(aKey)*.

Uit class *EPowerMap*:

```
public static RData GetValue(EPower aKey)
{
    return (!Value.m_SerializedT.Contains(aKey) ? null : Value[aKey]);
}

public static EPowerMap Value
{
    get
    {
        if (m_Mapper == null)
        {
            m_Mapper = new RData("02bf6f5a6ea7e0d4a903c7ff6b4c1165").target as EPowerMap;
        }
        return m_Mapper;
    }
}
```

In dit onderdeel wordt gecontroleerd of een *RData* object die bekend staat met identificatie "02bf6f5a6ea7e0d4a903c7ff6b4c1165" de specifieke waarde uit de enum *EPower* bevat.

```

public class RData : IEquatable<RData>, IOptzSerializable
{
    // Fields
    private Type m_CalculatedType;
    private RDataValue m_Data;
    private string m_IsAvailableErrorMessage;
    private InternalLogWrapper m_Log;
    public bool m_ShouldBeLoadedAsync;
    [SerializeField]
    private string RDataGUID;

    // Methods
    public RData();
    public RData(string guid);
    public RData(string aPath, string aName, string aType);
    public RData(string aPath, string aName, Type aType);
    public static bool ContainsData(RData aData);
    public void Dispose();
    public bool Equals(RData aOther);
    public override bool Equals(object aObject);
    public static RData FromPath(string aPath, Type aType);
    public override int GetHashCode();
    public static bool IsNullOrEmpty(RData aData);
    public Coroutine Load();
    public static bool operator ==(RData a, RData b);
    public static bool operator !=(RData a, RData b);
    IOptzSerializable IOptzSerializable.ReadBinary(BinaryReader aInput, int aVersion);
    void IOptzSerializable.WriteBinary(BinaryWriter aOutput);
    public override string ToString();

    // Properties
    public string AssetBundleName { get; }
    public int CurVersion { get; }
    private RDataValue Data { get; }
    public string GUID { get; }
    public bool IsAvailable { get; }
    public bool IsLoaded { get; }
    private InternalLogWrapper Log { get; }
    public string name { get; }
    public string path { get; }
    public string ResourcesName { get; }
    public Object target { get; }
    public Type type { get; }
}

```

In bovenstaande afbeelding is Public Class Rdata weergegeven. Dit is onderdeel van de Frima Engine library waar unity gebruik van maakt. Op basis van meerdere methods en properties kan worden geconcludeerd dat het gebruikt wordt om bestanden uit te lezen, namelijk:

- FromPath: Gebruik hiervan lijkt erop alsof er een bestandspad wordt bedoeld;
- ReadBinary: Lezen van binaire data;
- WriteBinary: Schrijven van binaire data.

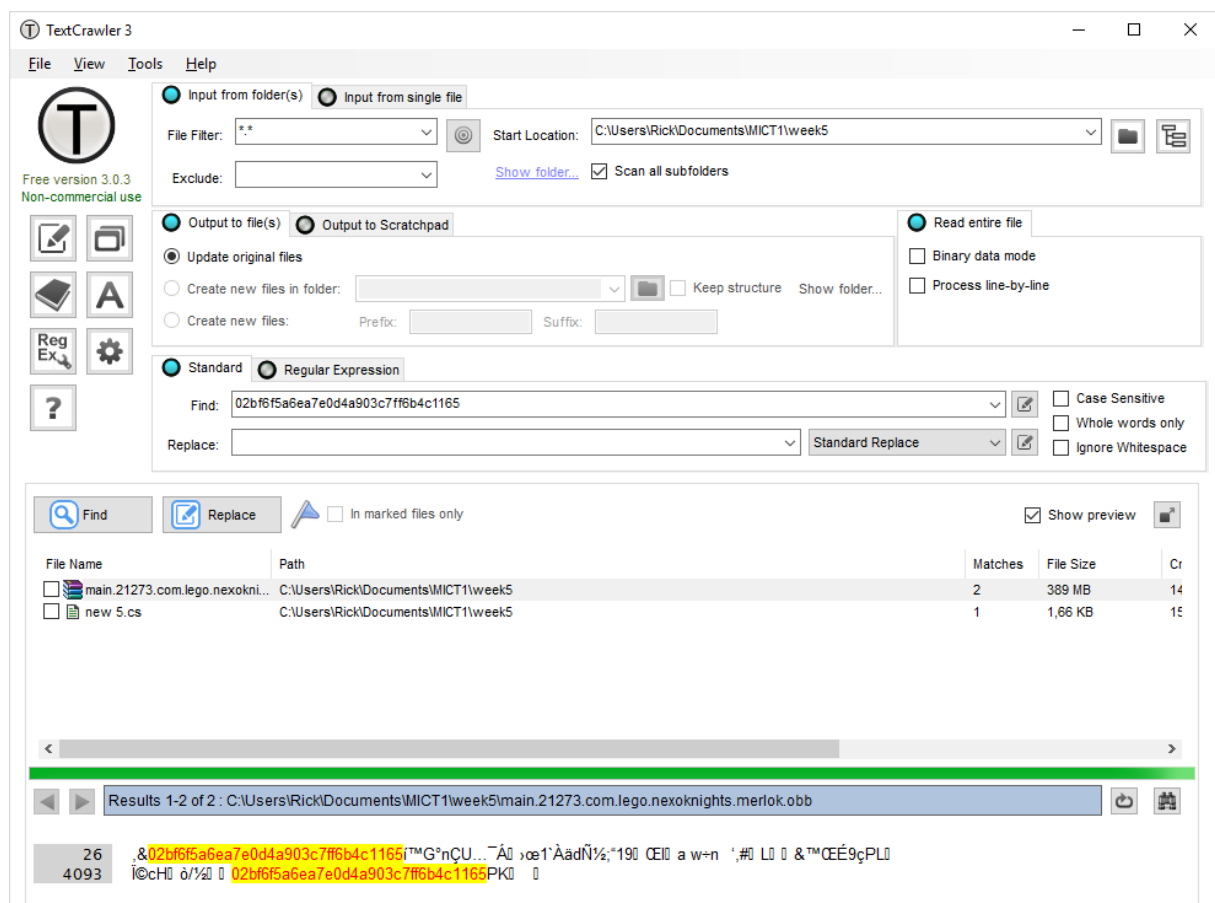
Bovenstaande handelingen worden vaak verricht in combinatie met een bestand.

Naar aanleiding van de specifieke identificatiestring is met behulp van het programma TextCrawler3 gezocht naar de waarde: 02bf6f5a6ea7e0d4a903c7ff6b4c1165 die in een bestand kunnen zitten. Voor dit onderzoek is het bestand “main.21273.com.lego.nexoknights.merlok.obb” gedownload van URL: <http://r6---sn-5hnedn7l.c.android.clients.google.com/market/GetBinary/GetBinary/com.lego.nexoknights.merlok/21273/o?mm=31&mn=sn-5hnedn7l&ms=au&mt=1457969856&mv=m&nh=lgpwZjAxLmFtczE1Kg4yMTMuNTEuMTU2LjlxMw&pl=14&expire=1458142770&ipbits=0&ip=0.0.0.0&sparams=expire,ipbits,ip,q:,mm,mn,ms,mv,nh,pl&signature=236D3A2199FEFA7F0EC8C95545C19A177E6E9F9A.AAA429CF33897A7136B8C262F5C7AA840BA1A5F&key=am3>

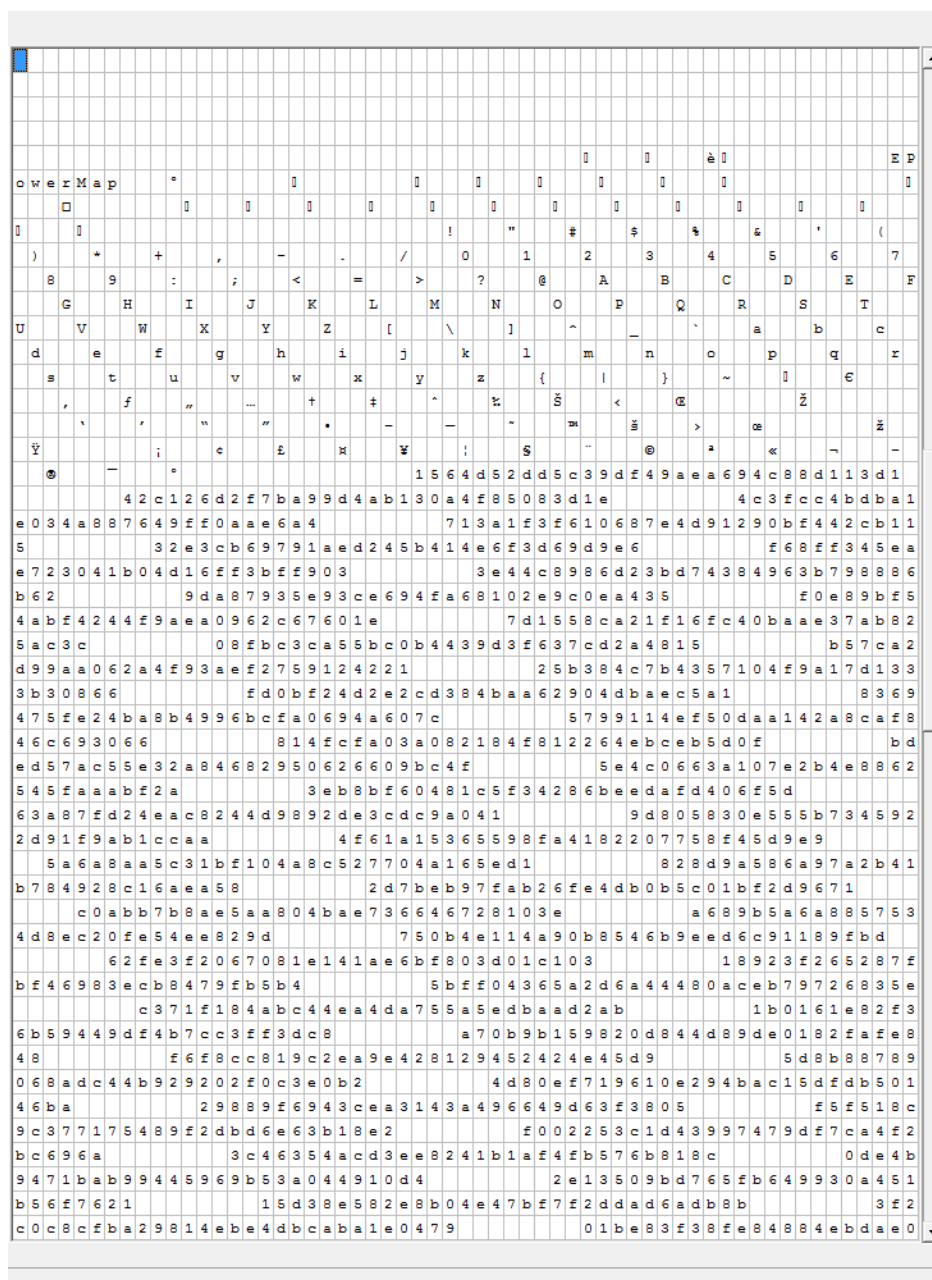
Dit bestand bevat niet de APK, maar alle additionele content die wordt gedownload door de applicatie tijdens het opstarten. Tijdens het controleren van het netwerkverkeer dat werd uitgevoerd door de applicatie met Fiddler is gebleken dat dit bestand alle additionele content bevat:

85	302	HTTP	android.clients.goo...	/market/download/AppDownload?packageName=com.lego.nexoknights.merlok&versionCode=21273&ft=o&token=AOTCm0TsHg5...	623	no-cache	text/html; c...
86	206	HTTP	r6---sn-5hnedn7l.c...	/market/GetBinary/GetBinary/com.lego.nexoknights.merlok/21273/o?mm=31&mn=sn-5hnedn7l&ms=au&mt=1457969856&mv=m...	301.7...	public, ...	application/...

Aan de hand van TextCrawler is één resultaat gevonden binnen dit bestand met additionele content:



In dit bestand bevindt zich een bestand die “02bf6f5a6ea7e0d4a903c7ff6b4c1165” wordt genoemd. Het script maakt van dit bestand de EPowerMap die gebruikt wordt als bron van data om de power te kunnen vinden.



In bovenstaande afbeelding is het bestand geopend in hex editor XVI32. In dit bestand zijn meer hash-achtige waarden te vinden die mogelijk ook refereren naar andere bestanden. Echter wordt eerst gekeken naar de broncode om te controleren of het bestand daadwerkelijk de waarden zoals te zien in enum *EPower* bevat.

```
public static RData GetValue(EPower aKey)
{
    return (!Value.m_SerializedT.Contains(aKey) ? null : Value[aKey]);
}
```

In bovenstaande functie *GetValue* wordt een extra waarde *m_SerializedT* weergegeven. Deze waarde komt uit class *EnumMapper* en representeert een lijst van *RData* waarden. Vermoedelijk wordt op dit punt het bestand ingelezen als *RData* object.

- De binaire data in het bestand komt overeen met 3E en verwijst naar een specifieke hash;
- De string waarde *GAUNTLET_OF_TRUTH* komt overeen met de ASCII data van het bestand en verwijst naar een specifieke hash.

[illegible]

- De volgorde van de identificatiestings (zoals 3E) komt overeen met de volgorde van de hashachtige string.

$$((4392 - 4144) / 4) + 1 = 63$$

3E is dus het 63^e karakter. Dit houdt in dat de bijbehorende hash de 63^e is in de reeks. Dit is waarde: d8359dfbd9ae28a48a11a9efe02dff4 (te vinden op positie: 0x1C88). Wederom wordt met behulp van TextCrawler gezocht naar de betreffende waarde. Deze waarde is te vinden in bestand AssetsBundle-Android.bytes. Deze is ook gedownload van URL: <http://lv-nexoknights-live-h.legocdn.com/live/1.1.1/AssetBundles/V23/AssetsBundle-Android.bytes> die bekend is geworden door het netwerkverkeer te onderscheppen.

In dit bestand staat de data opgenomen als (postie 0x16B5B9):

```
cKeyToken=null f002253c1d43997479df7ca4f2bc696a FrimaEn
gine.RDataValue, FrimaEngine.RData, Version=1.0.0.0, Cultur
e=neutral, PublicKeyToken=null Powers/FoulSteam FoulSteamDa
taUfoulSteamData, Assembly-CSharp, Version=0.0.0.0, Culture
=neutral, PublicKeyToken=null [FrimaEngine.RData, F
rimaEngine.RData, Version=1.0.0.0, Culture=neutral, PublicK
eyToken=null d83659dfbd9ae28a48a11a9efe02dffd4 FrimaEngi
ne.RDataValue, FrimaEngine.RData, Version=1.0.0.0, Culture=
neutral, PublicKeyToken=null Powers/GauntletOfTruth Gauntle
tOfTruthData[GauntletOfTruthData, Assembly-CSharp, Version=
0.0.0.0, Culture=neutral, PublicKeyToken=null [Frim
aEngine.RData, FrimaEngine.RData, Version=1.0.0.0, Culture=
```

Deze waarde is dus te koppelen aan de Gauntlet Of Truth. De hypotheses zijn dus waarschijnlijk waar:

- De binaire data in het bestand komt overeen met 3E en verwijst naar een specifieke hash;
- De volgorde van de identificatiestings (zoals 3E) komt overeen met de volgorde van de hash-achtige string.

In dit bestand worden verwijzingen gedaan naar de code (Assembly-Csharp). Echter verwijst het bestand verderop naar een AssetBundle die gerelateerd is aan de Gauntlet of Truth (positie 0x1B92DF).

```
ject, UnityEngine, Version=0.0.0.0, Culture=neutral, Public
KeyToken=null 2 Power/GauntletOfTruth/VO/en-GB-Android.a
ssetbundle v5 7a028de25ff44efa8947803fe0cd5945 j
```

Deze string is vergelijkbaar met de URL die eerder is gevonden bij het onderscheppen van het internetverkeer met de applicatie:

<http://lv-nexoknights-live-h.legocdn.com/live/1.1.1/AssetBundles/V17/Power/GauntletOfTruth/VO/nl-Android.assetbundle>

en

<http://lv-nexoknights-live-h.legocdn.com/live/1.1.1/AssetBundles/V17/Power/GauntletOfTruth/VO/nl-BE-Android.assetbundle>

Het is mogelijk dat de PowerData in deze assetbundles is opgenomen. Onder kopje: Analyseren van assetbundle is reeds een hypothese opgesteld:

- Deze power-gerelateerde assetbundles bevatten informatie die gebruikt wordt om het schild te identificeren na het scannen ervan;

Het is onbekend of in deze assetbundles informatie zit die gekoppeld kan worden aan het ShieldID. De hexadecimale waarden van het MarkerID zijn niet terug te vinden in de corresponderende assetbundle, zowel in little endian formaat als big endian formaat.

```

public static PowerData GetPowerData(EPower aKey)
{
    RData aData = GetValue(aKey);
    if (RData.IsNullOrEmpty(aData))
    {
        Debug.LogError("EPowerMap::GetPowerData. aKey: " + aKey.ToString());
        return null;
    }
    return (aData.target as PowerData);
}

```

Vervolgens wordt volgens bovenstaande code het bestand dat is geïdentificeerd via de mapping en als RData object wordt gebruikt, return gestuurd als PowerData. Het ongeïdentificeerde bestand is dus waarschijnlijk de bron van de ShieldBorderID.

Conclusie

De schilden worden gescand met het algoritme van Vuforia. Dit algoritme bevat een API, QCAR, waardoor het mogelijk wordt in een XML-bestand de instellingen om een object te herkennen door de camera vast te leggen. De instellingen die zijn aangetroffen bevatten:

- Bitlocations: Locaties waar de bits in de afbeelding kunnen voorkomen. Bits worden weergegeven in de vorm van vierkantjes;
- Bitradius: Grootte van de weergave van een bit (vierkantje);
- Blocklength: Hoeveel bits er geteld moeten worden;
- MessageLength: Potentiële grootte van de daadwerkelijke bitstring.

Uit de analyse van meerdere schilden is gebleken dat deze 0-en of 1-en in groepen van maximaal drie achter elkaar hebben als binaire identificatie. Daarnaast begint de identificatiestring altijd met 00010. Initieel was het aantal mogelijkheden schilden gelijk aan 4.294.967.296, maar dit is veel minder door deze beperkingen.

De analyse van het netwerkverkeer van de applicatie is uitgevoerd toen de app voor het eerst werd opgestart. Hieruit is gebleken dat er eerst een resource pack wordt gedownload met veel bestanden vanuit de google playstore. Daarna wordt een configuratie JSON gedownload die instellingen bevat zoals URLs die gebruikt kunnen worden om de rest van de content te downloaden. Vervolgens worden alle assetbundles gedownload van de servers die gedefinieerd stonden in het JSON-bestand. Bij deze assetbundles zitten ook power-gerelateerde assets. Het is echter onbekend of deze ook informatie zoals het ShieldID bevatten.

De analyse met .NETReflector heeft de volgende bevindingen naar voren gebracht:

- Enum EPower bevat alle namen van de schilden. Daarnaast bevat het de hexadecimale locaties van de schilden in het bestand "02bf6f5a6ea7e0d4a903c7ff6b4c1165";
- Het MarkerID is het resultaat van het scannen van de afbeelding, dit is een unsigned integer. Op basis van de frame marker beschrijving van Vuforia is geconcludeerd dat binaire data omgezet wordt naar deze integer. De rest van de handelingen wordt uitgevoerd aan de hand van deze uint.
- Powers die bij schilden horen worden opgehaald op basis van de enum EPower via de class EPowerMap;
- Class EPowerMap verwijst naar een RData object. Dit is een dataobject die binaire informatie over een bestand kan bevatten;
- In EPowerMap wordt verwezen naar RData object "02bf6f5a6ea7e0d4a903c7ff6b4c1165". In dit bestand staan referenties naar andere bestanden en is het letterlijk een map. Deze bestanden zijn tot op heden niet terug kunnen vinden, maar wel terug te koppelen aan het bijbehorende schild;
- De Powermapping in bestand 02bf6f5a6ea7e0d4a903c7ff6b4c1165 is gebaseerd op de volgorde van de waardes. In Enum EPower staat gedefinieerd welke hex-waarde hoort bij welk schild. In dit bestand staat gedefinieerd welke hex-waarde hoort bij welk bestand. De volgorde die wordt gehanteerd voor de hex-waardes is gelijk aan de volgorde die wordt gehanteerd voor de hash-achtige waardes.

Het is onbekend waar de bestanden die gedefinieerd staan in het bestand naartoe leiden. Het is echter wel bevestigd dat ze gerelateerd zijn aan de betreffende power die wordt opgevraagd. De mogelijkheid bestaat dat de gedownloadde assetbundles wel informatie zoals het shieldID bevatten.

Bronnen

1. Microsoft. (z.j.). Dictionary Class. Geraadpleegd van [https://msdn.microsoft.com/en-us/library/xfhwa508\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/xfhwa508(v=vs.110).aspx)
2. PKWare Inc.. (z.j.). ZIP File Format Specification. Geraadpleegd van <https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT>
3. PTC Inc.. (z.j.). Frame Markers. Geraadpleegd van <https://developer.vuforia.com/library/articles/Training/Frame-Markers-Guide>
4. Shieldpowerlist.com. (z.j.). 415-Gauntlet of Truth. Geraadpleegd van <http://www.shieldpowerlist.com/axl-shield-powers/415-gauntlet-of-truth/>
5. Unity Technologies. (z.j.). ScriptableObject. Geraadpleegd van <http://docs.unity3d.com/ScriptReference/ScriptableObject.html>