1 Inleiding

Op dinsdag 25 januari was ik na een nacht te lang coden het nieuws aan het napluizen en kwam ik op een bericht uit waar de *Tweakers.net* redactie het had over het publiceren van een eigen persbericht over de *cybercrimechallenge*. Dat vond ik natuurlijk een leuk gegeven, de KLPD zoekt slimme mensen. Op het zelfde moment keek ik ook in de webmail van onze stichting en zag ik dat een beleidsmedewerker van de Provincie Groningen een screenshot had gestuurd die mij deed watertanden, de dag kon niet beter beginnen dan met een mooie puzzel.

Ik bedacht me natuurlijk geen moment en begon de tekst eerst inhoudelijk te lezen. Wat is nu precies de bedoeling? Het werd me niet direct duidelijk. Wat me wel duidelijk was dat de opdracht waarschijnlijk zat in de analyse van een PCAP bestand.

2 Download de PCAP

Laten we er vanuit gaan dat niemand heeft afgehaakt bij stap 0, het openen van een Flash pagina. Natuurlijk is het van *Tweakers.net* super flauw een pagina in Flash te maken, als het geheel erg makkelijk in HTML5 gemaakt had kunnen worden. Maar goed: Download de PCAP.

PCAP is een formaat waarin netwerk pakketjes in zijn geheel worden opgeslagen. Het is wat PCM Wave is voor audio, alles zit er in het is: *lossless*. De standaard wordt in veel tools gebruikt. Denk hierbij aan het commandline *tcpdump* maar ook grafische programma's als *Wireshark* werken er eenvoudig mee. Laten we daar eens naar kijken.

Net als een goed boek, is deze handleiding geen stripverhaal. Met een contextuele beschrijving moet je in staat zijn de stappen die ik hieronder beschrijf te herhalen. En zelfs als een een gereedschap niet beschikbaar is voor jouw platform, is er vast wel iets anders te vinden dat wel werkt en ongeveer het zelfde doet. Mijn voorkeurskeuze gaat uit naar Wireshark op Linux.

3 Wireshark

Het PCAP formaat is niet al te complex om te lezen. Al zal de gemiddelde tekstverwerker er wel wat moeite mee hebben. Wireshark tovert een PCAP bestand om naar een grafische representatie. Ieder pakketje is een los regeltje, per pakketje kun je zien wat voor type pakketje het is.

Zoals je misschien weet is op de meeste netwerken de maximale lengte van zo'n pakketje 1500 bytes, verschillende pakketjes moet je dus aan elkaar plakken om een compleet bestand of webpagina over te sturen. Je hebt dus te maken met een hele boel pakketten en op het eerste oog zie je niet wat je nu met deze brei van informatie aan moet.

Gelukkig is Wireshark in staat om het netwerkverkeer ook nog op een andere manier te visualiseren. Door alle pakketjes en datastromen bij elkaar te vegen kan hij ook de gesprekken laten zien. Niet geheel onverwachts ben ik daar begonnen, met wie heeft onze grote boef gepraat? Door in Wireshark naar *Statistics, HTTP, Requests...* te gaan kun je alles netjes visualiseren. Wat ziet mijn oog daar? De meeste informatie die is verstopt komt vanaf een enkel IP adres. Als je goed oplet zul je ook zien dat dit IP adres in de zelfde reeks ligt als het bron IP adres van de rest van de pakketjes, vast het subnet van de VPN verbinding.

Op Twitter zag ik de afgelopen dagen dat mensen gingen proberen van dit IP adres direct wat te downloaden. Ik zou je ter overweging willen geven dat dit echt een compleet verkeerd pad is. Stel dit is voor het echie en een crimineel zie dat iemand anders bestanden download... u stapt hem? Daarbij weten we inmiddels ook dat dit een VPN verbinding was. Het is daarmee helemaal niet vanzelfsprekend dat dit IP adres überhaupt van buiten toegankelijk is. Het is daarom slimmer om het pakketje eens te bekijken en de flow te volgen.

Hoe dan? Als filter kun je opgeven: *ip.addr == 203.0.113.59*. Sorteer even op protocol dan zie je de HTTP requests boven aan staan. De twee downloads zijn nu eenvoudig te herkennen. Ik, in al mijn 'snelheid' en onwetendheid, heb via de rechtermuisknop de Flow gevolgd. En het stukje opgeslagen en de HTTP header in vim er bovenaf geknipt. Ik hoorde later op Twitter dat het veel eenvoudiger is om in Wireshark via *File, Export, Objects, HTTP* het bestand gewoon netjes op te slaan. Nou, dat weten we dan voor de volgende keer.

openkvk

Het is de #KLPD toch gelukt, door een leuk puzzeltje #cybercrimechallenge mij even bezighouden. Grasduinen blijft leuk.

4 De twee APK bestanden

Zoals je in Wireshark waarschijnlijk al zag aan *application/vnd.android.package-archive* hebben we hier te maken met 'iets voor Android'. Nu heb ik een Android telefoon, een hele oude die ik toch naar Ice Cream Sandwich wilde flas-

hen, maar ik heb het nooit zo op binaries van derden. En al helemaal niet als ze van de **KLPD** afkomen. Ik ben geen aluhoedje, maar is dit niet wat we elkaar de afgelopen jaren hebben geleerd? Vertrouw nooit onbekende binaries.

Maar goed, binaries dus en sourcecode, aldus de omschrijving. APK bestanden zijn eigenlijk gewoon zip bestanden. Oke, herstel, het zijn eigenlijk gewoon jar bestanden, die toevallig met *unzip* te openen zijn. Laten we dat dan maar gewoon doen he? Niet draaien op een telefoon, maar gewoon kijken wat er in zit.

Een bestand pakt uit, het tweede niet. Er zit een wachtwoord op. Ik geef toe: ik heb eerst 'Johnny' geprobeerd.

5 Encryptie op zip-bestanden

Encryptie op zip-bestanden is op een aantal manieren te kraken. Een methode is een bruteforce aanval. Je probeert gewoon heel veel combinaties en op een gegeven moment gaat het bestand open. Een zip-bestand is geen pinpas, je mag een onbeperkt aantal keer proberen. Voor dit soort 'uitdagingen' is *fcrackzip* beschikbaar. Mijn eee-pc was daarmee vrolijk aan de slag gegaan.

openkvk

Bruteforce ftw #cybercrimechallenge Real Cops Use Helicopters!

6 Gewoon doorgaan!

Real Cops never quit ;) Je kunt dit moment als rust beschouwen, of je gaat nog even grasduinen in de Wireshark, of je analyseert gewoon class bestanden met de hand. Dat kan natuurlijk met een decompiler. Ik wil op dit moment even stil staan bij Hanpeter van Vliet, hij is in 1996 gestorven aan kanker. Een van de dingen die hij heeft nagelaten is de Mocha Java decompiler. Dit pad heb ik niet gekozen, ik ben zelf met *strings* en *vim* aan de gang gegaan of ik wat kon vinden in de class-bestanden.

Omdat ik totaal niet wist hoelang het kraken kon gaan duren bedacht ik me het volgende: *"Op dit moment moeten er mensen nu aan de slag zijn die op hetzelfde niveau als ik bezig zijn, een echte agent werkt altijd samen, deelt competenties en expertise. Het beste voor de groep is het delen van kennis, je weet nooit hoe moeilijk het wordt."* De APKs heb ik naar mijn eigen webserver ge-upload, zodat meer mensen zouden kunnen gaan kraken. Ondertussen waren andere mensen op

twitter gekomen die vragen begonnen te stellen. Er was activiteit: samen hacken, altijd leuk.

openkvk

Aan collega's, Subj; #cybercrimechallenge #botnet; Potentiele bedreigingen in #VPN. stefan.konink.de/contrib/klpd/ Ons team kraakt verder.

7 Er was een wachtwoord

Omdat ik ongeduldig was had ik ook een Core i7 van Calendar42.com misbruikt. In hun eigen belang uiteraard, een dagje niet werken is ook zonde immers. (Sorry, Michel!) Maar het was allemaal niet nodig, m'n lieve kleine eee-pc kon het gemakkelijk aan en met de uitkomst *botnet* was ik in het geheel niet ontevreden. Bestand uitgepakt, ingepakt als tar.gz en weer geupload naar de server, zodat meer mensen weer verder kon gaan. Het zou nog wel eens pittig kunnen worden, zag ik al toen ik een blik nam op wat nog te doen stond.

openkvk

Aan collega's, Subj; #cybercrimechallenge #botnet; Succesvolle wachtwoordkraak. Ons team analyseert de broncode op eerdere locatie.

8 Anoniem blijven

Stel je werkt als buitengewoon opsporingsambtenaar en je bent lekker aan het grasduinen. Niet iedere crimineel hoeft te weten waar jij vandaan komt. Dat soort dingen wekken argwaan. We hebben al gezien op welk netwerk onze vriend Johnny zich bevond. Mochten we daar in de toekomst echt heen moeten surfen, dan is het verstandig om dat via een anonymiseringsdienst te doen, denk hierbij bijvoorbeeld aan *Tor*.

openkvk

Aan collega's, Subj; #cybercrimechallenge Herinnering, gebruik alleen ons anonieme #tor netwerk voor activiteiten richting 193.200.198.84.

Mij viel op dat in de broncode andere adressen stonden dan in de PCAP, dat soort dingen spreken mij dan aan. Er is vast meer te halen, het is een aanwijzing die moet worden onderzocht.

openkvk

Aan collega's, Subj; #cybercrimechallenge De broncode maakt melding van een ander IP dan verkeer in de #VPN. 193.200.198.133

Een andere goede bron van informatie is de WHOIS database. Wie zijn eigenlijk de eigenaren van de eerder genoemde IP adressen? En waar bevinden zij zich? Dat is simpel en snel uit te vinden. Waarom is dit relevant? Stel dat deze dump onvoldoende informatie had opgeleverd en er was wat aan de gang zou het wel handig zijn om de ISP te kunnen benaderen, om een nieuwe tap te plaatsen of klantgegevens op te vragen.

openkvk

Aan collega's #Haaglanden, Subj; #cybercrimechallenge #ovj #huiszoekingsbevel voorbereiden voor 2491AJ 130 #Pine #Digital Security

9 De analyse van de broncode

Omdat ik nog steeds geen executable heb gedraaid is de vraag natuurlijk: wat is dit voor applicatie en waar moet je dan beginnen in die broncode? Meestal ga ik af op header bestanden (voor het overzicht) of gewoon broncode lezen. In dit geval kunnen we gelukkig een aantal java-bestanden vinden in de broncode. Hadden we iets anders kunnen verwachten in een Android applicatie?

Zoals je net al zag had ik een URI gevonden met een IP adres. Grote kans dat deze URI lijdt naar een locatie waar het licht niet komt. Mijn Tor netwerk draaide inmiddels en ik ging eens kijken. Je moest een wachtwoord invullen... geheel onverwachts natuurlijk. In diezelfde broncode wordt melding gemaakt van een wachtwoord. Als je op *PASS* in de broncode gaat zoeken kom je ook tegen dat dit wachtwoord zelf met *Base64* is gecodeerd.

Base64 is een methode om een binair bericht in een zichtbare tekenreeks te coderen. Het is dus niet echt een versleuteling, maar meer een manier om berichten als platte tekst te versturen die eigenlijk binair van aard zijn. In dit geval moeten we dus het gecodeerde Base64 bericht omzetten naar het origineel. Niet geheel onverwachts; het origineel is helemaal geen binaire reeks. Mijn programmeertaal van keuze om snel dingetjes mee te experimenteren is

nu Python, je kunt daarmee zeer eenvoudig van een base64 bericht naar een normaal bericht komen.

```
import base64
print base64.decodestring('aGRHaXh1aUxWZTNRQWxxME9FSEFQQlAx')
```

Met dat wachtwoord kon ik het forum eenvoudig op. Ik zag op twitter voorbij komen dat mensen begonnen te zeuren over het feit dat ik netjes, in alle openheid, mijn voortgang deelde. Nouja, ik was misschien op het forum aangekomen. Maar was de volgende opdracht wel zo makkelijk? De berichten stonden versleuteld op het forum, dat zag er niet goed uit. Ik dacht terug aan mijn encryptie lessen van Peter den Brok. Falen was *geen* optie, dit moest ik kunnen anders kon ik mijn *ing.* titel wel schrappen. Vergezeld met een flinke dosis angstzweet begon ik de berichten een voor een in een tekst bestand te kopiëren. Ik zag op tijd dat ik niet de hele regel te pakken had met een standaard X11-kopieer-selectie.

openkvk

Aan collega's, Subj; #cybercrimechallenge bewijsmateriaal. Ontcijferen berichten. stefan.konink.de/contrib/klpd/c... #politie = #team pic.twitter.com/10tO295O

10 Crypto-analyse

Ik denk dat het beste is om samen eens door de Java code te zeven. Inmiddels is mij duidelijk dat de berichten van het forum worden opgehaald. En Johnny z'n lessen in het programmeren wel degelijk heeft gevolgd en zelfs commentaar heeft gebruikt om zijn overwegingen te documenteren: kuddo's voor jou. Het is altijd goed om je te verplaatsen in je opponent.

Hoe voor de hand liggend het allemaal ook is, in de message klasse gebeuren een aantal dingen. Het eerste ding wat er gebeurd is het bericht van een Base64 tekenreeks omtoveren in een binaire representatie. Erg slim, binaire teksten op een PHP forum levert bijna altijd ongewenste resultaten op. Of dat nu door de PHP of de MySQL komt, of misschien wel de HTTP Character Encoding, is een compleet ander onderwerp. Johnny begreep dat.

Om vervolgens dat *binaire bericht* te decrypten hebben we een sleutel nodig en een bit. Laat ik nu eerst vertellen hoe ik *dacht* dat het werkte:

Het algoritme werkt door per opvolgend bericht een sequentie in te bouwen die als salt werkt. Je moet de sequentie van een bericht weten voor je hem kunt decoderen. Een willekeurig bericht is dan compleet waardeloos omdat deze ontvangen is in de verkeerde volgorde.

Vandaag werd door Jobert Tabma op Twitter uitgelegd dat dit niet zo was. *Bit* wordt gebruikt voor een visuele animatie om een bericht te decoderen. De data wordt 'langzaam' gedecodeerd, maar is pas volledig zichtbaar als de loop klaar is. Goed om te weten, zeker als je de app niet aan hebt gehad. Later meer hier over.

Maar we zitten eigenlijk op een zijspoor. Het gaat er natuurlijk om wat dat wachtwoord wel niet is. Nu ligt het heel erg voor de hand als ik dit allemaal opschrijf, eenvoudig is het allemaal niet. Laten we net zoals bij iedere wiskunde som eens beschrijven wat de bekende en onbekende variabelen zijn:

- this.message, het bericht dat base64 gedecodeerd is
- password, onbekend
- bit, is altijd 7 voor succesvol decoderen
- de versleuteling loopt gelijk met het bericht en de sleutel
- de versleuteling herhaalt zich, als het bericht langer is dan de sleutel
- het gedecodeerde bericht begint altijd met ENCRYPTED

Waarom is dat laatste van zo'n enorm belang? Dit impliceert: **als we het woord ENCRYPTED kunnen ontsleutelen, hebben we het begin van de versleuteling gekraakt!** Een eventuele tekstuele analyse kunnen we daarna altijd nog doen om zo de rest van het wachtwoord te vinden.

Hoe werkt dat algoritme nu precies? Voor het aantal tekens dat het inmiddels binaire bericht lang is, kijken we naar de huidige positie in het bericht. Deze halen we op en is variabele d, de teken positie modulo de wachtwoord lengte resulteert in variabele c. Wat betekent dat laatste nu precies? Als het wachtwoord korter is, dan het bericht, beginnen we weer bij de eerste letter van het wachtwoord. Om tot het resultaat teken te komen wordt $c\ xor\ d$ uitgevoerd. Voor wie xor niet kent: het betekent 'enkel of'. Binair gezien laadt het dus alleen een *waar,onwaar* door, maar nooit *onwaar,onwaar* of *waar,waar*.

Na heel veel gepuzzel van mensen weten we inmiddels dat bit 7 moet zijn, maar het betekent niet dat de iteraties daarvoor overgeslagen kunnen worden, immers: anders zouden we alleen de achtste bit decrypten. We zullen dus of acht keer door het bericht heen moeten decrypten, of we nemen een kortere weg: we passen het algoritme aan en slaan die regel helemaal over. Op dat laatste kwam ik na wat debuggen uit en heeft me best veel tijd gekost. Voordat ik deze regel wiste pakte ik de eerste letter en rekende ik met de hand uit wat het moest zijn en zag dat het dan nooit uit kon komen.

```java
public void decrypt(String password, int bit) {
    String result = "";
    int i;

    if (password.length() == 0) {
        this.message = this.COULD_NOT_DECRYPT_NO_PASSWORD;
        return;
    }

    for (i=0; i<message.length(); i++) {
        int c = password.charAt(i % password.length());
        int d = message.charAt(i);

        c = c & (1 << bit);

        result = result + (char)(c ^ d);
    }

    if (bit == 7 && !result.startsWith("ENCRYPTED")) {
        this.message = this.COULD_NOT_DECRYPT_INVALID_KEY;
    } else {
        this.message = result;
    }
    return;
}
```

11 Bruteforce aanval, of toch niet

We weten inmiddels dat het begin van de berichten altijd met ENCRYTPED begint. Niet zo handig van Johnny. Meer mensen op Twitter viel al op dat alle berichten met hetzelfde versleuteling beginnen. Dit moet je automatisch opvallen, impliciet is dit een een zwakte in elke encryptie.

Even een klein stukje goede encryptie: begin eerst met het verhogen van de informatie dichtheid, de entropie. Denk hierbij aan het 'verzippen' van het bericht, als iets een kleiner volume heeft, en het zelfde gewicht is de dichtheid hoger. Je weet inmiddels dat het dan niet handig is als je een bijvoorbeeld een header ervoor laat staat, maar het globale idee is duidelijk.

Wij weten dat Johnny niet bij Peter den Brok in de klas heeft gezeten. En

we weten ook dat als het ons zou lukken om de eerste 9 karakters goed te decoderen dat we er al zijn. Mooi: dat scheelt een hoop tijd. Stukje code dan maar?

```
def decrypt(msg, bit = 1):
    out = ''
    for i in range(0, 9):
        d = ord(msg[i])
        current = ord('ENCRYPTED'[i]);
#         print 'letter', d
#         print 'encrypted', current


        for c in range(32,177):
#           c = c & (1 << bit); # niet gebruiken dus
            if (c ^ d) == current:
                out += chr(c)
                continue;
```

Zoals je ziet is dit bijna het algoritme van Java in Python. Ik ga voor een willekeurig bericht de eerste 9 tekens af. Als eerste haal ik uit het woord ENCRYPTED de numerieke representatie van de letter. Zeg maar het nummer wat je in de ASCII tabel zou vinden, daar kun je makkelijker mee rekenen. Wat debugging om m'n leven makkelijker te maken door de letters uit te printen.

Je ziet nu ook dat ik ga slechts de tekens 32 to 177 ga doorzoeken. Dit zijn spatie tot del, uit de ASCII tabel. Heb je meer nodig? Ik denk het in een typische situatie niet. Vervolgens ga ik voor de huidige letter kijken of de berekeningen $c\ xor\ d$ de huidige letter wordt. Is dat zo sla ik hem op en ga ik verder.

Wanneer je deze code gebruikt zul je zien dat hieruit wel een wachtwoord komt, maar er iets spannends gebeurt als je dit wachtwoord zou gebruiken om de berichten te decoderen. Het bericht klopt maar gedeeltelijk. Als je dan nog een keer naar het wachtwoord kijkt, zul je zien dat het zich herhaalt. Nu moet je een stukje teruglezen: *wanneer herhaalde het wachtwoord zich?*

openkvk

Aan collega's, Subj; #cybercrimechallenge onze cryptoanalysten hebben het begin van de #sleutel gevonden. stefan.konink.de/contrib/klpd/a...

Het wachtwoord herhaalt zich op het moment dat het bericht langer is dan het wachtwoord. De gedachte dat het wachtwoord langer zou zijn is dus niet waar, het tegenovergestelde is zelfs waar. De sleutel om het bericht te decoderen is zelfs korter dan het woord wat wij al weten. Door dit wachtwoord in te vullen in de eigen implementatie van het algoritme, kunnen we de berichten van het forum een voor een lezen.

12 Conclusie

Het was een leuke uitdaging om eens lekker te rotzooien met cryptografie. Praktisch nut: geen. Ik heb namelijk niet de illusie dat er een recruiter van de KLPD op mijn opgegeven telefoonnummer gaat bellen. Hoe had deze challenge spannender gemaakt kunnen worden? Ik ben een groot voorstander van SSL Decryptie. Dat is inmiddels een hobby van me geworden en ik kan het iedereen aanraden. Maar goed, dat is niet realistisch en zou vrijwel iedereen al laten afvallen op het moment van de PCAP analyse. Wat wel spannender was geweest was het verlengen van het wachtwoord, zodat je echte tekstuele analyse had moeten doen. Dat zie je wel eens bij de CSI series, ik dacht zelf meer aan 'Viper'. Die series spelen zich overigens ook buiten af, naar een fysieke locatie moeten is dan ook wel eens leuk: kom je ook nog buiten!

Een ander onderdeel wat ik heb gemist is het echte samenwerken. Niet iedereen kan goed zijn in alle onderdelen van het politiewerk. Iedereen heeft zijn beperkingen en daarom is de combinatie van het beste in jou en de beste in een ander de echte meerwaarde voor een Hightech Crime Team. Het motto *durf te vragen* wordt naar mijn mening genadeloos afgestraft op *Gathering of Tweakers*. Het beste voorbeeld is wel dat aldaar wordt gepredikt om eerst te Googlen. Leg je op Twitter al je handel en wandel uit en ben je daarmee 'googlebaar' word je door de *crew* uitgemaakt voor *spoiler*. Zoek een ander forum of medium waar je samen tot iets moois kan komen, waarbij moderatie niet als excuus wordt gebruikt om het niveau hoog te houden.

MAVZRCJbD0xUBSdcRkxUFilXHgldRDMcYGZgECNUCwI=