Teoria e Prática de Ataques de Segurança

CC4029

Security Tokens (App Anda)

Realizado por:
António Cunha (up201805142)
Cláudia Maia (up201905492)

Conteúdo

1 Introdução **2**
1.1 Diferentes tipos de tokens 2
1.2 Caso de estudo 3
1.2.1 Como funciona? 4
1.2.2 Diferentes tipos de título metro do Porto 4

2 learning Process **4**

3 Análise **5**
3.1 Como funciona a App (no que toca ao NFC) 5

4 Vulnerabilidades **6**

5 Conclusão **7**

1 Introdução

No âmbito deste projeto, escolhemos o tema de *Security Tokens*, com foco em *tokens NFC*. Mas antes de entrar em detalhe neste subtópico é importante definir e ter uma breve visão sobre *security tokens*.
Um *security token* é um dispositivo periférico utilizado para ganhar acesso a um recurso eletronicamente restrito, pode ser utilizado em conjunto com outros mecanismos de autenticação como *passwords* ou códigos *pin* e atua como uma chave eletrónica para aceder a algo, ou desempenhar uma função como abrir portas, aceder a uma conta bancária online, entre outros...
Habitualmente, os *security tokens* costumam conter material criptográfico como chaves que podem ser utilizadas, por exemplo, na geração de assinaturas digitais. Assim sendo, podemos afirmar que todos os *tokens* contém algum tipo de informação secreta necessária numa prova de identidade. Existindo 4 formas distintas desta informação ser utilizada:

- **Password Estática:** O aparelho contém a password que está escondida e invisível ao proprietário, sendo transmitida a cada autenticação[1];
- **Passwords síncronas dinâmicas:** É utilizado um temporizador para percorrer combinações produzidas por um algoritmo criptográfico, isto obriga a que tanto o *token* como o servidor tenham relógios sincronizados;
- **Passwords assíncronas:** Gera uma *one-time password* sem a utilização de um relógio mas sim com um algoritmo criptográfico (como um one-time-pad);
- **Desafio-Resposta:** Utilizando uma chave criptográfica para provar posse da mesma sem a revelar (resolvendo um desafio relativo à chave em questão);

No nosso projeto, olhámos para a tecnologia NFC, uma tecnologia que utiliza **passwords estáticas**, na subsecção seguinte iremos abordar alguns dos tokens de password estática e como é que se comparam com o NFC.

1.1 Diferentes tipos de tokens

Existem diferentes tipos de tokens, sendo os 3 principais:

- **Connected tokens:** Neste tipo de tokens os utilizadores precisam de conectar fisicamente o token ao sistema, para que este seja detetado. Um exemplo deste tipo de token são os smartcards, já que é necessário passar o cartão no leitor para que a informação de autenticação chegue ao sistema.
- **Disconnected tokens:** Neste caso, não é necessário inserir fisicamente o token em leitores para fazer a leitura do token, no entanto pode ser necessário introduzir um código gerado pelo token. As autenticações do telemóvel por duplo fator são um exemplo deste tipo de token, já que

[1] O que torna este tipo de *tokens* vulneráveis a ataques de repetição

é gerado um código apresentado no telemóvel para fazer a autenticação geralmente.

- **Contactless tokens:** Este tipo de tokens não requerem inserir fisicamente o token ou introduzir um código gerado, em vez disso, estes dispositivos conectam-se com o sistema sem fios e o acesso é garantido ou recusado baseado na conexão. Assim os Bluetooth tokens e NFCs tokens são exemplos desta tecnologia.

NFC Near Field Communication é um tecnologia sem fios que permite conexões de curta distância entre dois dispositivos compatíveis, tipicamente uma NFC tag e um smartphone ou tablet. As NFC tags são pequenos chips que armazenam dados e são geralmente incorporados em stickers, ímanes ou etiquetas. Esta tecnologia requer que os utilizadores apenas aproximem o dispositivo do leitor já que opera e, curtas distâncias de menos de 10 cm.

RFID Radio Frequency Identification, tal como o nome indica, é um método para identificar itens apenas por ondas radio. Tal como o NFC funciona sem fios e transfere data sem precisar de contacto físico (contactless). esta tecnologia usa tags ou cartões para armazenar os dados e não precisam de ser carregada, ou seja, não precisa de energia. Um sistema RFID é constituído pela tag, um leitor e uma antena. As RFID tags podem ser ativas ou passivas, sendo que as ativas contêm uma fonte de energia para que possam fazer broadcast para ler numa distância de até 100 metros. Para além disso estas podem ser Low Frequency (LF, 125 – 134 kHz, com range < 0.5m), High Frequency (HF, 13.56 MHz, com range < 1.5 m), Ultra High Frequency (UHF, 865 – 956 MHz, com range $0.5\text{m} < \text{d} < 5\text{m}$) e Super High Frequency (SHF, 2.45 GHz, com range $<$ 10 m).

Assim, as RFID tags Low frequency podem ser EM4100 (usadas maioritariamente como cartões de identificação ou de acesso), T55x7 (usadas para emular outros cartões de Low Frequency) ou HID. Geralmente, estes tipos de cartões são mais vulneráveis pelas falhas de segurança e há falhas na deteção de ataques.

Do mesmo modo, as RFID tags High Frequency podem ser Mifare 1K/4K, Mifare Ultralight, Mifare Ultralight C, Mifare DESFIRE, ou Mifare PLUS. Os Mifare cards cumprem o padrão internacional 14443A, desenvolvido pela International Organization for Standardization (ISO). Este padrão 14443A é constituído por 4 partes, sendo estas: Características físicas (1); Potência de radiofrequência e interface de sinal (2); Inicialização e anticolisão (3); e por fim o protocolo de transmissão (4). Os Mifare 1k/4k, Ultralight/C usam o padrão ISO/IEC 14443 até à parte 3, enquanto os Mifare DESFIRE, PLUS usam o padrão na totalidade.

1.2 Caso de estudo

O nosso caso de estudo foi então a aplicação Anda, a solução dos Transportes Intermodais do Porto que surgiu em 2018 para facilitar a deslocação nos trans-

portes do Porto, ao permitir utilizar o telemóvel como um título de viagem ao emular um cartão NFC (NXP - Mifare Plus) de modo a substituir o título físico tradicional.

1.2.1 Como funciona?

Esta aplicação tem como objetivo permitir viajar nos transportes públicos do Grande Porto aderentes ao sistema intermodal Andante de forma simples, rápida e cómoda. A vantagem oferecida pela aplicação, em relação aos cartões físicos, é viajar sem precisar de conhecer todas as regras tarifárias, tendo como garantia pagar o menor valor possível pelas viagens realizadas, no final do mês. Esta aplicação pelo que percebemos, utiliza informações de localização e a velocidade de deslocação de modo a saber o meio de transporte que se está a utilizar assim saber quando terminou a viagem (uma vez que nos transportes do Porto apenas é necessário passar o cartão emulado no início da viagem e não no fim) isto permite assim à aplicação fazer o cálculo do menor valor possível a cobrar aos clientes tendo em conta as zonas e o tempo de viagem. Para além destas informações a aplicação utiliza também um sistema de *Beacons* de modo a confirmar a localização real do smartphone, sem que esta seja controlada pelo utilizador. Os Beacons são pequenos dispositivos que utilizam a tecnologia *Bluetooth Low Energy (BLE)*. Esta emite um sinal intermitente de ondas de rádio e assim consegue localizar o smartphone num determinado raio.

1.2.2 Diferentes tipos de título metro do Porto

Após explorar os diferentes tipos de cartão de títulos do metro do Porto com a aplicação *NFC tools* verificamos que o andante de papel é do tipo Mifare Ultralight EV1, o passe mensal é ISO 14443-4 (High frequency card) e que o cartão emulado pela aplicação *Anda* é NXP - Mifare Plus, ISO 14443-4.

2 learning Process

Instalação da App e Descompilação: Para podermos fazer reverse engeneering da App *Anda*, instalamos a aplicação para android através do site APK-PURE, de seguida descompilamos o ficheiro *.apk* com a ajuda do *JadxGUI*, onde consiguimos exportar projeto como *projeto gradle* e analisar depois no *IntelliJ IDEA*, já que o *JadxGUI* não nos permite mudar o nome das variáveis, para facilitar na análise do código.

Pesquisa: O processo de pesquisa esteve sempre de "mãos dadas"com o processo de análise do código, pois dado o elevado nível de ofuscação do mesmo, a maior parte da informação obtida foi graças à pesquisa de bilbiotecas utilizadas e de API's de Java necessárias na emulação e gestão de cartões NFC para android.

3 Análise

3.1 Como funciona a App (no que toca ao NFC)

com.spirtch.android.hce.calypso.ContactlessService: Após uma analise detalhada das várias classes em java, conseguimos perceber que para emular o cartão NFC, a aplicação Anda utiliza a classe *android.nfc.cardemulation.HostApduService* extendendo a classe *HostApduService* como ContactlessService e fazendo override de certos atributos e funções da classe do java. Com base na documentação do java, esta classe emula cartões baseados no protocolo NFC-Forum ISO-DEP, que por sua vez são baseados no protocolo ISO/IEC 14443-4 e suportam processing command Application Protocol Data Units (APDUs) como definidas na especificação ISO/IEC 7816-4. Como verificamos anteriormente com o leitor NFC tools, o cartão emulado pela aplicacao é de facto ISO 14443-4 e tem ainda como tecnologia disponível ISO-DEP.

```
1 import android.annotation.TargetApi;
2 import android.content.Intent;
3 import android.nfc.cardemulation.HostApduService;
4 import android.os.Bundle;
5 public class ContactlessService extends HostApduService {
```

Excerto Relevante na identificação da classe de emulação ContactlessService

p205h.p206a.p207a.AndaHCEManager: Para além da parte da emulação dos cartões percebemos ainda que esta classe AndaHCEManager é onde é gerida toda a informação dos cartões como o timestamp, a zona, a linha, a localização e entre outras informações da viagem. Isto porque HCE refere-se ao Host Card Emolation, ou seja, gere as informações daquela viagem em concrete, que está a decorrer.

```
1  import pt.card4b.backendbeacons.App;
2  import pt.card4b.backendbeacons.DataModels;
3
4
5  public class AndaHCEManager {
6              ...
7              ...
8      public void notifyAnswer(int i, int i2, JSONObject jSONObject)
       {
9              ...
10             ...
11             try {
12                                 sb.append("System date: ");
13                                 sb.append(GeneralUtils.m1593b(
       SysDateManager.m1497d().m1495f()));
14                                 sb.append(" Event data - timestamp:
       ");
15                                 sb.append(GeneralUtils.m1593b(
       GeneralUtils.m1602a(timeInMillis)));
16                                 sb.append(" provider:");
17                                 sb.append(parseLong);
```

```
                                    sb.append(" line:");
                                    sb.append(parseLong3);
                                    sb.append(" zone:");
                                    sb.append(parseLong5);
                                    sb.append(" location:");
                                    sb.append(parseLong2);
                                    sb.append(" direction: ");
                                    sb.append(j);
                                    sb.append(" variant:");
                                    sb.append(j2);
                                    try {
                                        AppLog.m2158a("LOGIC_HCE", "
    AndaHCEManager", "Validation notify", sb.toString());
                                    } catch (Throwable th3) {
                                        th = th3;
                                    }
```

Card4b: Assim verificamos que para a recolha e tratamento de dados a app Anda recorreu aos serviços da Card4b, uma empresa de software portuguesa, sediada em Matosinhos, que oferece serviços de Integrated Mobility solutions e city-services, para interoperável contactless ticketing e entre outros serviços. Com isto percebemos que a card4b oferece também apoio backend no acesso às bases de dados, ajudando assim a app a perceber o trajeto que o utilizador fez, assim como quando terminou a viagem.

4 Vulnerabilidades

No que toca a vulnerabilidades, a nossa abordagem inicial foi pesquisar brechas na aplicação e relatórios de falhas e/ou *exploits* detetados. Foi aqui que nos deparámos com o artigo do Gustavo Silva (disponível aqui) que aproveita o facto de alguns dos métodos utilizados na aplicação não serem implementados mas sim diretamente importados como uma biblioteca nativa compilada da **JNI**(**J**ava **N**ative **I**nterface) para conseguir interagir com a base de dados da mesma sem qualquer tipo de privilégios, obtendo informações sensíveis sobre os utilizadores como o **Nome**, **Morada**, **Email**, **Nº Telemóvel**.
Esta falha, encontrada em 2018 foi reportada e corrigida, no entanto ao analizar o código descompilado reparámos em alguns padrões como certos métodos, como o **verifyBeacon** que retornam simplesmente uma chamada a uma outra função estática e nativa que aceita os mesmos argumentos. Notámos ainda que a *native-lib* de Java continua a ser carregada na aplicação, pelo que suspeitámos que a correção na falha supramencionada, baseou-se apenas numa perspetiva de "segurança por ofuscação", estratégia pouco segura. No entanto não nos foi possível montar um *proof-of-concept*, tendo ficado pela teoria.

5 Conclusão

Em tom de conclusão, conseguimos identificar como a aplicação funciona, localizando em específico a zona do código onde este cartão é emulado. No entanto não nos foi possível clonar o cartão, dado o funcionamento da aplicação, o cartão em questão não tem dados de viagem mas apenas de perfil. Além do mais este *token* necessita de localização GPS, pelo que, clonar para uma tag NFC seria ilógico e levaria a um não funcionamento na utilização da aplicação, já que os dados da viagem utilizados na cobrança são recolhidos ao longo da viagem através dos *beacons* e localização supramencionados. Assim sendo, um bom ponto de partida para comprometer a integridade da aplicação seria através de *reverse engineering* do código, explorando a vulnerabilidade mencionada acima e não tanto a componente de NFC por si só.

Referências

[1] "Security token," Wikipedia, 30-Jan-2023. [Online]. Available: here. [Accessed: 31-Jan-2023].

[2] "Beacon: O que É e como eles podem ser úteis? • usemobile," Usemobile, 06-Dec-2022. [Online]. Available: here. [Accessed: 31-Jan-2023].

[3] What is a security token (or authentication token) and how does it work?," Okta. [Online]. Available: here. [Accessed: 31-Jan-2023]

[4] "RFID vs NFC - what's the difference?," Wireless Links, 11-Jun-2022. [Online]. Available: here. [Accessed: 31-Jan-2023].

[5] "Systems S.A.," Card4B. [Online]. Available: here. [Accessed: 31-Jan-2023]

[6] "Hostapduservice; android developers," Android Developers. [Online]. Available: here. [Accessed: 31-Jan-2023].

[7] "Host card emulation (HCE)," Thales. [Online]. Available: here. [Accessed: 31-Jan-2023].