0.1 Анализ безопасности

Не стоит переоценивать данную технологию и забывать о таком важном аспекте, как безопасность. На текущий момент развития трафик в процессе передачи никаким образом не шифруется, поэтому злоумышленник потенциально может провести атаку.

Выделяются 2 главных типов атак: спуффинг и пиггибэкинг [1].

- *Beacon Spoofing.* Атака основана на включении маячка или устройства, работающего как маячок, с параметрами, идентичными параметрам некоторой группы маячков. Это может быть использовано для того, чтобы симулировать некоторое событие или оповещение в месте, отличном от предполагаемого в контексте приложения.
  Пример угрозы: предположим, есть некоторый магазин, использующий маячки для приветствия новых посетителей. Злоумышленник копирует настройки маячков, и впоследствие это позволяет ему разослать приветственное уведомление пользователям, которые находятся в абсолютно другой локации. Это может спровоцировать недоумение и, в конечном итоге, неудовлетворенность программным продуктом.

- *Beacon Piggybacking, или Hijacking.* Атака основана на применении уже существующих маячков, предназначенных для некоторого приложения, в своем, стороннем приложении. Это может быть использовано для получении аналитики, основанной на оригинальный маячках, а также в рассылке сообщений и симуляции событий, не относящихся к предполагаемым.
  Пример угрозы: пусть кофейня «А» конкурирует с кофейней «Б». «А» начинает использовать приложение с использованием маячков. В ответ кофейня «Б» разрабатывает свое приложение с информацией об идентификаторах чужих маячков. В результате каждый раз, как пользователь с установленным приложением от кофейни «Б» заходит в кофейню «А», на его устройство приходит оповещение о скидках на кофе в «Б».

Неприятный инцидент произошел на выставке потребительской электронике (Consumer Electronics Show – CES) в 2014 году. Для всех желающих была

организована «охота за сокровищами», в рамках которой через приложение пользователи должны были определить место, где якобы зарыт сундук с кладом. Однако еще до начала мероприятия неизвестная группа хакеров взяла apk-файл приложения, проанализировала его структуру через декомпилятор, и смогла извлечь параметры всех используемых маячков. Это давало всю необходимую информацию, и победить таким способом можно было даже не выходя из дома [2].

Выделяют 4 способа защиты:

1. **Геолокационная проверка**. После получения оповещения от одного из новых (в рамках текущей сессии) маячков, устройство использует геолокационный сервис, чтобы убедиться, что маячок физически действительно находится поблизости.

2. **Идентификация на основе начального значения (seed)**. Используются маячки с периодически меняющимся UUID. Алгоритм смены, в свою очередь, основан на некотором цифровом значении, хранимом отдельно. Через единый SDK происходит обновление, идентификация и синхронизация всего процесса. Маячок, не прошедший проверку – потенциальный вредитель – будет выкинут из рассмотрения.

3. **Облачное подтверждение**. Способ базируется на основе предыдущего, но роль связующего звена на себя берет облачный сервис, а не локальный SDK. Этот механизм уже используется в маячках компании Estimote, и известен под названием “UUID Rotation”. Секретный ключ, определяющий смену идентификаторов, хранится на платформе Estimote Cloud.

4. **Управление на уровне аппаратных средств**. Начальные параметры и их возможное обновление берут на себя аппаратные средства – контроллеры. При этом оповещение об обновлении приходит от облачного сервиса отдельно на контроллеры и отдельно на устройство пользователя. После этого контроллеры обновляют UUID маяков. В дальнейшем проверка маячков будет происходить на устройстве без использования сервиса.

Преимущества и недостатки перечисленных методов представлены в таблицах 1 и 2, соответственно.

Способ защиты	**Геолокационная проверка**	**Идентификация на основе генерации случайных чисел**	**Облачное подтверждение**	**Управление на уровне аппаратных средств**
Преимущества	1. Наиболее дешевый способ 2. Простой в конфигурации и поддержке 3. Может быть легко включен или выключен в любой момент 4. Позволяет использовать традиционный iBeacon-формат для лучшей совместимости	1. Позволяет защититься от перечисленных типов атак 2. Нет привязки к геолокации или Интернет-соединению 3. Не требует дополнительного оборудования	1. Позволяет в большей степени защититься от перечисленнных способов атак 2. Не требуется подключение дополнительных устройств 3. Усложняет процедуру проведения атаки для злоумышленника	1. Надежный уровень защиты 2. Предоставляет устройство для обновления и обслуживания маячков 3. Изменения легко применить к существующей системе 4. Позволяет использовать традиционный iBeacon-формат для лучшей совместимости

Способ защиты	**Геолокационная проверка**	**Идентификация на основе генерации случайных чисел**	**Облачное подтверждение**	**Управление на уровне аппаратных средств**
Недостатки	1. Не защищает против piggybacking-атак 2. Геолокационная относительность не дает такой же точности, как остальные методы 3. Определение локации может являться причиной задержки в начале сессии	1. Сложен в модификации в случае неполадок 2. UUID маячков могут быть легко определены злоумышленником 3. Более сложное развертывание 4. Формат использования не поддерживается в приложениях Apple	1. Сложен в модификации в случае неполадок 2. Сложен в развертывании 3. Задержки в использовании могут отразиться на удобстве использовании приложения 4. Предполагает наличие Интернет-соединения 5. Формат использования не поддерживается в приложениях Apple	1. Наиболее дорогой тип размещения 2. Требует наличия дополнительного набора устройств 3. Не работает для маячков, расположенных на удалении от остальных 4. Предполагает наличие Интернет-соединения (периодически)

Примечательно, что Apple iOS SDK на уровне реализации не позволяет приложению сканировать эфир на обнаружение BLE-пакетов. Обязательным условием является явная конфигурация UUID, major и minor-идентификаторов [3].

Список литературы

[1] Secure beacons: Overview & options. — URL: http://www.slideshare.net/localzco/beacon-security-overview. — 2015. — (дата обращения: 20.04.15г.).

[2] Hacking the ces scavenger hunt. — URL: http://makezine.com/2014/01/03/hacking-the-ces-scavenger-hunt/. — 2014. — (дата обращения: 04.04.15г.).

[3] Corebluetooth doesn't let you see beacons. — URL: http://developer.radiusnetworks.com/2013/10/21/corebluetooth-doesnt-let-you-see-ibeacons.html. — 2013. — (дата обращения: 22.03.15г.).

Приложения

Приложение 1. Реализация алгоритма, основанного на поиске силового центра

```
double * calculateUserPositionPowerCenter(double *xs, double *ys, double *
      accs) {

    double *result = (double *)malloc(sizeof(double) * DIMENSIONS);

    // k[n] = ( B[n].x^2 + B[n].y^2 - dist[n]^2 )/2
    double *ks = (double *)malloc(sizeof(double) * MIN_BEACONS);
    for (size_t i = 0; i < MIN_BEACONS; ++i) {
        ks[i] = (xs[i]*xs[i] + ys[i]*ys[i] - accs[i]*accs[i])/2.0;
    }

    // D = | x1-x2  y1-y2 | = (x1-x2)*(y2-y3) - (y1-y2)*(x2-x3)
    //     | x2-x3  y2-y3 |
    double d = (xs[0] - xs[1])*(ys[1] - ys[2]) - (ys[0] - ys[1])*(xs[1] - xs
          [2]);

    if (fabs(d) < 1e-7) {
        result[0] = -1;
        return result;
    }

    // X = | k1-k2  y1-y2 | = (k1-k2)*(y2-y3) - (y1-y2)*(k2-k3)
    //     | k2-k3  y2-y3 |
    double x = (ks[0] - ks[1])*(ys[1] - ys[2]) - (ys[0] - ys[1])*(ks[1] - ks
          [2]);

    // Y = | x1-x2  k1-k2 | = (x1-x2)*(k2-k3) - (k1-k2)*(x2-x3)
    //     | x2-x3  k2-k3 |
    double y = (xs[0] - xs[1])*(ks[1] - ks[2]) - (ks[0] - ks[1])*(xs[1] - xs
          [2]);

    result[0] = x/d;
    result[1] = y/d;

    return result;
}
```

Приложение 2. Реализация алгоритма, основанного на пересечении сфер

```
double * calculateUserPositionSphereIntersection(double *xs, double *ys,
      double *accs) {
    double *result = (double *)malloc(sizeof(double) * DIMENSIONS);

    double temp = (xs[1] - xs[0])*(xs[1] - xs[0]) + (ys[1] - ys[0])*(ys[1] -
          ys[0]);
    double exx = (xs[1] - xs[0]) / sqrt(temp);
    double exy = (ys[1] - ys[0]) / sqrt(temp);

    double p3p1x = xs[2] - xs[0];
    double p3p1y = ys[2] - xs[0];

    double ival = exx * p3p1x + exy*p3p1y;

    double p3p1i = (xs[2] - xs[0] - exx)*(xs[2] - xs[0] - exx) + (ys[2] - ys
          [0] - exy)*(ys[2] - ys[0] - exy);

    double eyx = (xs[2] - xs[0] - exx) / sqrt(p3p1i);
    double eyy = (ys[2] - ys[0] - exy) / sqrt(p3p1i);

    double d = sqrt(temp);

    double jval = (eyx * p3p1x) + (eyy * p3p1y);

    double xval = (accs[0]*accs[0] - accs[1]*accs[1] + d*d) / (2*d);
    double yval = (accs[0]*accs[0] - accs[2]*accs[2] + ival*ival + jval*jval
          )/(2*jval) - (ival/jval)*xval;

    result[0] = xs[0] + exx*xval + eyx*yval;
    result[1] = ys[0] + exy*xval + eyy*yval;

    return result;
}
```

Приложение 3. Реализация адаптивного геометрического алгоритма

```
#define MAX_ITERATIONS  500
#define Points          struct point *
#define enoughPoints    7
#define step            0.1f
```

```
/**
 *  Struct represents a beacon or a point
 */
struct point {
    double x;
    double y;
    double r;
};

/**
 *  Checks whether the point belongs to all three circles
 *
 *  @param x      x
 *  @param y      y
 *  @param points Beacons
 *
 *  @return 1 if true, 0 otherwise
 */
int isPointBelongToAllCircles(double x, double y, Points points) {
    int belongs = 1;
    for (size_t i = 0; i < MIN_BEACONS; ++i) {
        double dist = getDistance(x, y, points[i].x, points[i].y);
        if (dist > (points[i].r + 1e-2)) {
            belongs = 0;
            break;
        }
    }
    return belongs;
}

/**
 *  Returns the centroid coordinate of given points
 *
 *  @param points Array of points
 *  @param size   Its size
 *
 *  @return Centroid coordinates
 */
Points getCenter(Points points, size_t size) {
    Points center = (Points )malloc(sizeof(struct point));
    center->x = 0;
    center->y = 0;

    for (size_t i = 0; i < size; ++i) {
        center->x += points[i].x;
```

```
        center->y += points[i].y;
    }
    center->x /= size;
    center->y /= size;

    return center;
}

/**
 *  Calculates all circle-circle intersections and return an array of
        resulting point
 *
 *  @param pointA Circle A
 *  @param pointB Circle B
 *  @param cnt    Reference to the resulting array's size
 *
 *  @return Array of points
 */
Points getCircleCircleIntersection(Points pointA, Points pointB, int *cnt) {
    *cnt = 0;
    double r1 = pointA->r;
    double r2 = pointB->r;
    double p1x = pointA->x;
    double p1y = pointA->y;
    double p2x = pointB->x;
    double p2y = pointB->y;
    double d = getDistance(p1x, p1y, p2x, p2y);

    // if too far away, or self contained - can't be done
    if ((d >= (r1 + r2)) || (d <= fabs(r1 - r2))) {
        return 0;
    }

    double a = (r1*r1 - r2*r2 + d*d)/(2*d);
    double h = sqrt(r1*r1 - a*a);
    double x0 = p1x + a*(p2x - p1x)/d;
    double y0 = p1y + a*(p2y - p1y)/d;
    double rx = -(p2y - p1y)*(h/d);
    double ry = -(p2x - p1x)*(h/d);

    *cnt = 2;
    Points result = (Points )malloc(*cnt * sizeof(struct point));
    result[0].x = x0 + rx;
    result[0].y = y0 - ry;
    result[1].x = x0 - rx;
    result[1].y = y0 + ry;
```

```
    return result;
}

/**
 *  Returns an array of circle intersections
 *
 *  @param points Beacons
 *  @param size   Size of the array
 *  @param cnt    Reference to the size of the resulting array
 *
 *  @return Array of points
 */
Points getIntersectionPoints(Points points, size_t size, int *cnt) {
    size_t e = 20;
    Points result = (Points )malloc(e * sizeof(struct point));

    size_t indexToAdd = 0;
    for (size_t i = 0; i < size; ++i) {
        for (size_t j = 1; j < size; ++j) {
            int cnt = 0;
            Points intersects = getCircleCircleIntersection(&points[i], &
                  points[j], &cnt);
            if (cnt > 0) {
                for (size_t k = 0; k < cnt; ++k) {
                    result[indexToAdd++] = intersects[k];
                }
            }
        }
    }

    return result;
}

/**
 *  Selects only common points from the circles intersection
 *
 *  @param points Intersection points
 *  @param size   Size of that array *  @param beacons The given beacons and
       their accuracies
 *  @param cnt    Reference to the resulting array's count
 *
 *  @return Array of points
 */
Points getCommonPoints(Points points, size_t size, Points beacons, int *cnt)
      {
    Points result = (Points)malloc(sizeof(struct point) * enoughPoints);
```

```
    int k = 0;
    for (size_t i = 0; i < size; ++i) {
        if (!isPointBelongToAllCircles(points[i].x, points[i].y, beacons)) {
            continue;
        }

        result[k].x = points[i].x;
        result[k].y = points[i].y;
        result[k].r = points[i].r;
        ++k;
    }
    *cnt = k;

    return result;
}

/**
 *  Mutates three arrays of coordinates into one array of points
 *
 *  @param xs   xs
 *  @param ys   ys
 *  @param accs accs
 *
 *  @return Array of points
 */
Points createPoints(double *xs, double *ys, double *accs) {
    Points points = (Points )malloc(MIN_BEACONS * sizeof(struct point));
    for (size_t i = 0; i < MIN_BEACONS; ++i) {
        points[i].x = xs[i];
        points[i].y = ys[i];
        points[i].r = accs[i];
    }
    return points;
}

/**
 *  Enlarges the beacons' accuracies proportionally
 *
 *  @param beacons Array of beacons
 */
void enlargeAccuracies(Points beacons) {
    for (size_t i = 0; i < MIN_BEACONS; ++i) {
        beacons[i].r *= (1 + step);
    }
}
```

```
//*********************************************************************

double * calculateUserPositionEpta(double *xs, double *ys, double *accs) {
    double *result = (double *)malloc(sizeof(double) * DIMENSIONS);

    Points beacons = createPoints(xs, ys, accs);

    int iterations = 0;
    while (1) {
        if (++iterations > MAX_ITERATIONS) {
            return result;
        }

        int intersectionCount = 0;
        Points intersections = getIntersectionPoints(beacons, MIN_BEACONS, &
              intersectionCount);

        if (intersectionCount == 0) {
            return result;
        }

        int commonCount = 0;
        Points common = getCommonPoints(intersections, intersectionCount,
              beacons, &commonCount);

        if (commonCount == 2 || commonCount == 3) {
            Points center = getCenter(common, commonCount);
            result[0] = center[0].x;
            result[1] = center[0].y;
            return result;
        }

        enlargeAccuracies(beacons);
    }

    return result;
}
```

Приложение 4. Начальная настройка подключенной библиотеки

```
#import <BeaconLocation/BeaconLocation.h>
@import CoreGraphics;
// ...
@property (nonatomic, strong) BeaconLocation *library;
```

```
// ...
@interface MyClass: NSObject <BeaconLocationDelegate>
// ...
- (void)init {
  self = [super init];
  if (self) {
    //init
    _library = [[BeaconLocation alloc] initWithUUIDString:@"
          12345678-1234-0000-4321-876543210000"
                                                identifier:@"My region"];

    //beacons
    [_library.floor addBeaconWithMajor:0 minor:0 inPosition:CGPointMake(1,
          2)];
    [_library.floor addBeaconWithMajor:0 minor:1 inPosition:CGPointMake(2,
          3)];
    [_library.floor addBeaconWithMajor:0 minor:2 inPosition:CGPointMake(1,
          3)];

    //method
    [_library.processor setAlgorithm:AlgorithmTypeSphereIntersection];

    //delegate
    _library.delegate = self;
  }
  return self;
}

- (void)onUpdateUserPostion:(CGPoint)position {
  // do fancy stuff
}
```