Разработка метода генерации некорректных данных для проведения фаззинг-тестирования приложений с помощью NFC

Самарин Николай Николаевич
кандидат технических наук, начальник научно-исследовательского отделения № 6, ФГУП «НИИ «Квант», г. Москва, Россия, samarin_nik@mail.ru

АННОТАЦИЯ

Введение: В настоящей статье предложен метод генерации некорректных данных для проведения фаззинг-тестирования приложений с помощью NFC. В ходе исследований рассмотрен протокол сообщений NDEF, используемый при передаче данных с помощью NFC. На основе анализа протокола определены возможные конфликты и противоречия, которые можно внести в сообщения NDEF. Описаны функции, создающие различные конфликты в сообщениях NDEF, а также алгоритм генерации последовательностей некорректных сообщений с помощью данных функций. Предполагается продолжить проведение дальнейших исследований с целью разработки интеллектуального метода оценки покрытия тестируемого по NFC приложения, что позволит внести в разработанный метод модификации для повышения эффективности фаззинг тестирования.

КЛЮЧЕВЫЕ СЛОВА: информационная безопасность; фаззинг-тестирование; NFC; NDEF.

Введение

NFC (Near Field Communication) – это технология беспроводной передачи данных на короткие расстояния (до 10 сантиметров). Для выполнения передачи данных с помощью NFC используются специальные чипы, работа которых основана на электромагнитной индукции. Данная технология передачи может применяться, к примеру, для эмуляции банковских карт (оплата с помощью смартфона), для осуществления пропускного контроля через системы контроля и управления доступом (СКУД), для считывания данных с внешней аппаратной метки (NFC-метки), для передачи данных между мобильными приложениями.

В то же время на российском рынке увеличивается доля смартфонов со встроенным чипом NFC. Так, в 2022 году доля устройств с чипами Near Field Communication среди смартфонов, представленных на российском рынке, составила 70%, рис. 1 [1].

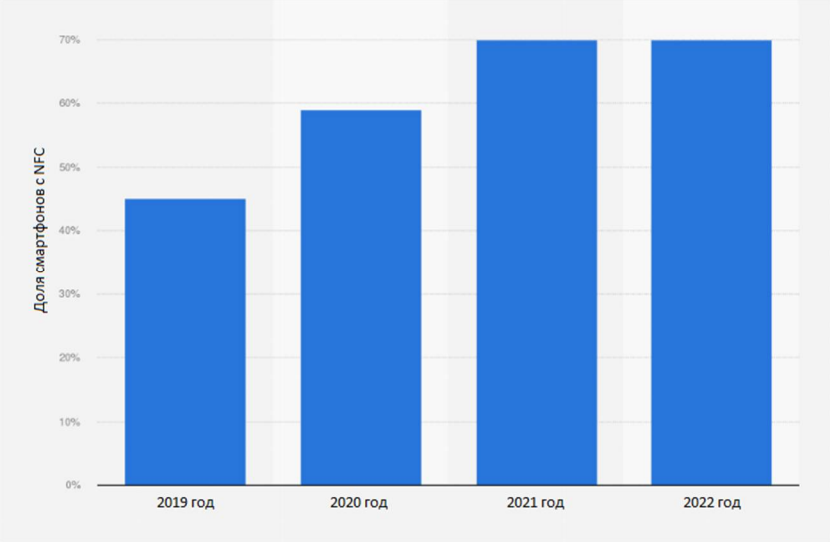

Рис. 1. Доля смартфонов с чипами NFC на российском рынке

Помимо указанных возможностей, использование технологии NFC также может приводить к нарушению безопасности устройств и приложений из-за эксплуатации уязвимостей, которые основаны на ошибках в реализации обработчиков пакетов NFC. К примеру, уязвимость CVE-2023-35671 позволяет злоумышленнику получить данные о сохраненных в «GoogleWallet» банковских картах с помощью внешнего считывателя NFC [2].

Для поиска и устранения ошибок в программном обеспечении может использоваться популярный подход в виде фаззинг-тестирования [3]. При проведении фаззинг-тестирования специальный программный или программно-аппаратный компонент (так называемый фаззер) передает на вход тестируемому приложению заведомо некорректные данные. Для формирования некорректных данных, как правило, используются две основные техники –мутация данных и генерация данных. При использовании первой техники в существующие данные вносятся случайные или заданные правилами изменения. При использовании второй техники некорректные данные генерируются заранее на основе протоколов или правил.

Наиболее актуальными работами в области фаззинга NFC являются [4] и [5].

Авторы работы [4] разработали алгоритм генерации тестовых сообщений «NFC Data Exchange Format» (NDEF), используемых для передачи данных с помощью NFC. Алгоритм использует четыре стратегии для построения тестовых примеров – вручную, генерация, мутация и «обратный анализ». На основе алгоритма также разработан

инструмент «GNFCVulFinder», предназначенный для поиска уязвимостей в приложениях Android и Windows.

Авторы работы [5] подробно исследуют стек протоколов NFC, форматы передаваемых данных. Описываются способы проведения фаззинг-тестирования различных уровней стека протоколов.

Несмотря на обширность исследований, проведенных в [4] и [5], предложенные авторами подходы к генерации некорректных данных обладают существенным недостатком, а именно, отсутствием возможности автоматической генерации тестовых данных с помощью внесения конфликтов в заголовки пакетов NDEF.

В связи с этим, в рамках настоящей работы рассматриваются особенности фаззинг-тестирования мобильных приложений с помощью сообщений NFC. Также приводится описание разработанного метода формирования некорректных входных данных для проведения фаззинг-тестирования с помощью NFC.

Передача данных с помощью NFC

При работе с NFC приложения, как правило, используют сообщения NDEF для обмена данными в двоичном формате. Основной контейнер данных, определенный форматом NDEF, называется NDEF сообщением, которое состоит из одной или нескольких записей NDEF разных типов. Тип указывает вид данных, которые содержат запись, а последовательность типов записей в сообщении определяет вид сообщения. Например, сообщение URI содержит одну запись, которая кодирует строку URL.

Для работы с сообщениями NDEF в мобильном приложении разработчики могут использовать как собственные, так и сторонние программные библиотеки, к примеру, набор инструментов разработки nRFConnect SDK [6]. Также существуют библиотеки для работы с определенными типами сообщений и записей, и универсальный генератор, который можно использовать для простой реализации других стандартизированных записей и сообщений, или даже для создания собственных записей.

Каждое сообщение NDEF содержит одну или несколько записей, которые состоят из заголовка и полезной нагрузки [7]. Заголовок записи содержит метаданные о типе и длине полезной нагрузки. Полезная нагрузка записи представляет собой ее фактическое содержимое. Структура сообщения и записи NDEF представлена на рис. 2.

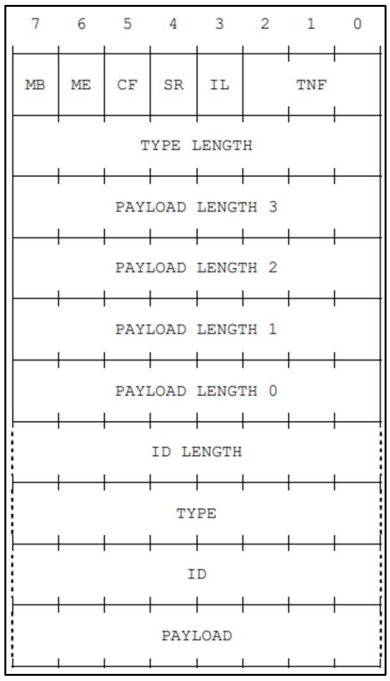

Рис. 2. Структура сообщения NDEF

Рассмотрим подробнее представленный формат NDEF.

Заголовок записи NDEF состоит из полей [6], представленных в табл. 1.

Табл. 1. Поля заголовка записи NDEF

Имя поля	Длина (в байтах)	Обязательное поле	Описание
Flagsand TNF	1	Да	Содержит флаги MB, ME, CF, SR, IL и TNF
TypeLength	1	Да	Указывает длину поля типа полезной нагрузки. Обязательно, но может быть равно нулю
PayloadLength	1 или 4	Да	Указывает длину полезной нагрузки. Длина может составлять 1 байт или 4 байта в зависимости от флага SR. Обязательно, но может быть равно нулю
ID Length	1	Нет	Требуется, если установлен флаг IL. Указывает размер поля идентификатора полезной нагрузки
Type	Различная	Нет	Требуется, если длина типа больше 0. Указывает тип полезной нагрузки записи NDEF
ID	Различная	Нет	Требуется, если установлен флаг IL и длина идентификатора больше 0. Указывает идентификатор полезной нагрузки записи NDEF. Может содержаться только в сообщениях с флагом MB

Рассмотрим подробнее флаги в поле «Flagsand TNF» [7].

Флаги MB (MessageBegin) и ME (MessageEnd) указывают положение записи NDEF в сообщении, если сообщение было фрагментировано на несколько записей. Флаг MB устанавливается для первой записи в сообщении, а флаг ME устанавливается для последней записи. Если запись является единственной записью в сообщении, то в ее заголовке устанавливаются оба флага.

Флаг CF (ChunkFlag) используется для фрагментированной полезной нагрузки. Устанавливается во всех фрагментах записи, за исключением последнего.

Флаг SR (ShortRecord) используется для определения размера поля длины полезной нагрузки. Если флаг установлен, длина полезной нагрузки должна занимать 1 байт, в противном случае длина равна 4 байтам.

Флаг IL (ID Lengthpresent) указывает, присутствует ли поле длины идентификатора в заголовке.

Флаг TNF (TypeNameFormat) определяет структуру поля типа полезной нагрузки и способ его интерпретации. Допустимые значения представлены в табл. 2.

Полезная нагрузка сообщения содержит информацию в произвольном формате, который определяется метаданными в заголовке сообщения. Каждый TNF имеет свои собственные типы полезной нагрузки и форматирование. При этом специальное значение TNF 0x05 позволяет указать собственное форматирование для личного использования. Для дальнейшего исследования будет рассматриваться тип «NFC Forumwell-knowntype», так как данный формат подробно описан стандартами.

Формат «NFC Forumwell-knowntype» содержит следующие типы полезной нагрузки:

тип T, обозначает текстовую запись [10];

тип U, обозначаетURI, используется для ссылок на веб-ресурсы [11];

тип Sp(SmartPoster), описывает мета-обертки, содержащие несколько сообщений, каждое со своей полезной нагрузкой [12].

Табл. 2. Возможные значения TNF

Код байта	СоответствующеезначениеTNF
0x00	Пустое поле
0x01	NFC Forum well-known type
0x02	Media-type, в соответствии с [8]
0x03	Absolute URI, в соответствии с [9]
0x04	NFC Forum external type
0x05	Unknown
0x06	Unchanged (используется при передаче сообщения частями)
0x07	Зарезервированное значение

Рассмотрим подробнее различные типы полезной нагрузки.

Текстовые записи (тип T) состоят из метаданных о кодировке текста и самого текста. При этом записи состоят из следующих трех полей:

Status Byte – занимает 1 байт;

ISO/IANA language code – занимает 2 или 5 байт;

Text – размер зависит от поля «Payload Length».

«Status Byte» указывает тип кодировки (UTF-8 или UTF-16) и длину языкового кода:

бит 7: значение 0 – кодировка текста UTF-8, значение 1 – кодировка текста UTF-16;

бит 6: зарезервирован для использования в будущем и всегда должен быть равен 0;

биты с 5 по 0: обозначают «Length Language Code», указывающий размер поля «Language Code» в байтах.

ISO/IANA language code – это языковой код документа (ISO/IANA), к примеру, кодами являются «en» для английского языка Великобритании, «en-US» для английского языка США, «ru» для русского языка и т.д.

Text – это область, которая содержит текст, кодировка и язык которого указываются в «Status Byte» и «ISO/IANA language code» [12].

Пример сообщения типа Text представлен в листинге 1.

Лист. 1. Пример сообщения типа Text

```
_____________________________
|       Message Header          |
|           0x11                | Status Byte,
|                               |   SR = 1 (Short Record, payload len =
1 byte),
|                               |   TNF = 0x01 (NFC Well Known Type),
|                               |   IL = 0, thus no ID Length of ID
field.
|           0x01                | Type Length = 1
|           0x0F                | PayloadLength = 15 Bytes (0x0F == 15)
|           0x54                | Type 'T' for Text Record
|-------------------------------|
|       Message Payload         |
|           0x02                | Text Record Status Byte -
|                               |   UTF flag  = 0 (message is UTF-8),
|                               |   Length of language code = 2 bytes
|           0x65                | = 'e' Language Code (2 Bytes) 'en'
|           0x6e                | = 'n'     for English
|           0x54                | = 'T' Text Begin
```

```
|             0x65              | = 'e'
|             0x73              | = 's'
|             0x74              | = 't'
|             0x20              | = ' '
|             0x6D              | = 'm'
|             0x65              | = 'e'
|             0x73              | = 's'
|             0x73              | = 's'
|             0x61              | = 'a'
|             0x67              | = 'g'
|             0x65              | = 'e' Text End
|-------------------------------|
```

Записи URI состоят из полей «Identifier code», размером в 1 байт, и «URI field». Идентификационный код используется в качестве сокращения для часто используемых адресов, чтобы уменьшить размер записи URI [11].

«Identifier code» добавляется к строке UTF-8 для создания, передаваемого URL. Допустимые значения поля «Identifier code» представлены в таблице 3.

Табл. 3. Допустимые значения Identifier code

Десятичное значение байта	**Шестнадцатеричное значение байта**	**Обозначение протокола**
0	0x00	Пустое значение
1	0x01	http://www.
2	0x02	https://www.
3	0x03	http://
4	0x04	https://
5	0x05	tel:
6	0x06	mailto:
7	0x07	ftp://anonymous:anonymous@
8	0x08	ftp://ftp.
9	0x09	ftps://
10	0x0A	sftp://
11	0x0B	smb://
12	0x0C	nfs://
13	0x0D	ftp://
14	0x0E	dav://
15	0x0F	news:
16	0x10	telnet://
17	0x11	imap:
18	0x12	rtsp://
19	0x13	urn:
20	0x14	pop:
21	0x15	sip:
22	0x16	sips:
23	0x17	tftp:
24	0x18	btspp://
25	0x19	btl2cap://
26	0x1A	btgoep://
27	0x1B	tcpobex://
28	0x1C	irdaobex://
29	0x1D	file://

Десятичное значение байта	Шестнадцатеричное значение байта	Обозначение протокола
30	0x1E	urn:epc:id:
31	0x1F	urn:epc:tag:
32	0x20	urn:epc:pat:
33	0x21	urn:epc:raw:
34	0x22	urn:epc:
35	0x23	urn:nfc:

Поле «URI field» представляет собой строку в кодировке UTF-8, содержащую фактическую строку для URL-адреса.

Пример сообщения типа URI представлен в листинге 2.

Лист. 2.Пример сообщения типа URI

```
_____________________________
|       Message Header          |
|           0x11                | Status Byte,
|                               |   SR = 1 (Short Record, payload len =
1 byte),
|                               |   TNF = 0x01 (NFC Well Known Type),
|                               |   IL = 0, thus no ID Length of ID
field.
|           0x01                | Type Length = 1
|           0x06                | PayloadLength = 6 Bytes (0x06 == 6)
|           0x55                | Type 'U' for URI Record
|-------------------------------|
|       Message Payload         |
|           0x04                | URI ID Code, 'https://'
|           0x79                | = 'y' URI Begin
|           0x61                | = 'a'
|           0x2e                | = '.'
|           0x72                | = 'r'
|           0x75                | = 'u' URI End
|-------------------------------|
```

SmartPoster – это особый вид сообщения NDEF, представляющий собой оболочку для других типов сообщений .«SmartPoster» обязан содержать поле «URI record» в единственном экземпляре, являющейся основной записью. Помимо «URI record» сообщение «SmartPoster» может содержать следующие поля [12]:

Titlerecord (экземпляры типа «Textrecord», их может быть несколько на нескольких языках);

Actionrecord – указывает действие, которое необходимо выполнить с URL (открыть – 0, сохранить – 1, отредактировать – 2);

Iconrecord – запись изображения типа «NDEF MIME»;

Sizerecord – используется для определения размера внешнего объекта, на который ссылается поле URI;

Typerecord – объявляет MIME-тип внешнего объекта, на который ссылается поле URI;

множество других типов записей, например, vCard и т.д.

Пример сообщения «SmartPoster» представлен в табл. 4.

Табл. 4. Пример сообщения «SmartPoster»

Сдвиг	Байтовое или строковое представление	Длина (в байтах)	Значение
0	0xD1	1	NDEF header. TNF = 0x01 (Well Known Type). SR=1, MB=1, ME=1
1	0x02	1	Длина имени записи SmartPoster (2 байта)
2	0x49	1	Длина полезной нагрузкиSmartPoster (73 байта)
3	“Sp”	2	Имя записи
5	0x81	1	NDEF заголовок. TNF = 0x01, SR=0, MB=1, ME=0
6	0x01	1	Длина имени записи (1 байт)
7	0x00, 0x00, 0x00, 0x0E	4	Длина полезной нагрузки URI (14 байт)
11	“U”	1	Тип записи: “U”
12	0x01	1	“http://www.”
13	“nfc-forum.org”	13	Непосредственно значение URI
26	0x11	1	Заголовок NDEFзаписи (SR=1, TNF=0x01)
27	0x03	1	ДлинаименизаписиThe length of the record name
28	0x01	1	Длина полезной нагрузки«act»
29	“act”	3	Тип записи: “act”
32	0x00	1	Action = Launchbrowser
33	0x11	1	Заголовок NDEFзаписи (SR=1, TNF=0x01)
34	0x01	1	Длина имени записи
35	0x12	1	Длина полезной нагрузки записи (18 байт)
36	“T”	1	Тип записи: “T” (=Text)
37	0x05	1	Status byte длязаписи Text (UTF8)
38	“en-US”	5	Код языка ISO: USEnglish
43	“Hello, world”	12	Непосредственно текст: “Helloworld” (в кодировке UTF-8)
55	0x51	1	Заголовок NDEFзаписи (SR=1, TNF= 0x01, ME=1)
56	0x01	1	Длина имени записи
57	0x13	1	Длина полезной нагрузки (19 байт)
58	“T”	1	Имя записи: “T”
59	0x02	1	Status byte: UTF-8
60	“fi”	2	Код языка ISO: Finnish
62	“Morjens, maailma”	16	Непосредственно текст: “Morjens, maailma” (в кодировке UTF-8)

Разработка метода генерации некорректных данных для проведения фаззинг-тестирования

В ходе анализа протокола сообщений NDEF обнаружено, что протокол вносит ряд определенных правил соответствия заголовков и ожидаемых значений пакета или последовательности пакетов, к примеру, сообщение с флагом MB должно идти перед сообщением с флагом ME [13-18].

Следовательно, для проведения фаззинг-тестирования приложений, использующих сообщения NDEF, возможно использовать подготовленные сообщения или группы сообщений, содержащие такие конфликты, которые нарушают рассмотренный протокол [19-25].

На основе анализа всех возможных конфликтов в сообщениях NDEF разработаны описания функций генерации последовательностей сообщений для проведения фаззинг-тестирования.Рассмотрим данные функции.

Создание общих конфликтов в сообщениях NDEF

Функция формирования сообщений в неполном виде (f1). На вход функции подается корректное сообщение, содержащее флаги MB и ME. Выходом функции является одно или несколько различных сообщений, у которых присутствует только первая часть сообщения (удаляется несколько последних байт).

Функция формирования сообщений без установленного флага MB (f2). На вход подается корректное сообщение, содержащее флаги MB и ME. Выходом функции является некорректное сообщение без флага MB.

Функция формирования сообщений без установленного флага ME (f3). На вход подается корректное сообщение, содержащее флаги MB и ME. Выходом функции является некорректное сообщение без флага ME.

Функция формирования пересекающихся сообщений (f4). На вход подается два корректных сообщения, содержащих флаги MB и ME. Выходом функции является некорректное сообщение, в котором флаг MB второго сообщения указан до появления флага ME первого сообщения (в теле пакета создается ситуация, когда два флага MB идут друг за другом, и между ними отсутствует флаг ME).

Создание конфликтов в фрагментированных сообщениях

Функция формирования сообщений с фрагментированной полезной нагрузкой в различных кодировках (f5). На вход подается последовательность сообщений, передающих части заданной полезной нагрузки в заданной кодировке. Выходом функции является некорректная последовательность сообщений, в которой у всех NDEF пакетов, кроме первого, в полезной нагрузке изменена кодировка на некорректную.

Функция формирования сообщений с фрагментированной полезной нагрузкой без установки флага CF в первом сообщении (f6). На вход подается последовательность сообщений с корректно расставленными флагами CF, передающих части заданной полезной нагрузки. Выходом функции является некорректная последовательность сообщений, в которой у первого сообщения отсутствует флаг CF.

Функция формирования сообщений с фрагментированной полезной нагрузкой различных типов (f7). На вход подается две последовательности сообщений, содержащих фрагментированные полезные нагрузки различных типов одинакового размера. Выходом функции является некорректная последовательность сообщений, первый пакет которой содержит тип данных, указанный в первой переданной на вход последовательности, а остальные содержат данные из второй последовательности.

Функция формирования сообщений с фрагментированной полезной нагрузкой с некорректно указанной длиной полезной нагрузки (f8). На вход подается последовательность сообщений с корректно расставленными значениями PAYLOAD_LENGTH, передающих части заданной полезной нагрузки. Выходом функции является некорректная последовательность, в сообщениях которой выставлены некорректные значения полей PAYLOAD_LENGTH (изменение значений у начального, промежуточных и последнего сообщений).

Функция формирования сообщений с фрагментированной полезной нагрузкой без установки флага CF на промежуточные сообщения (f9). На вход подается последовательность сообщений с корректно расставленными флагами CF, передающих части заданной полезной нагрузки. Выходом функции является некорректная

последовательность сообщений, в которой у нескольких промежуточных сообщений отсутствует флаг CF.

Функция формирования сообщений с фрагментированной полезной нагрузкой, конечное и промежуточные сообщения которой имеют значения полей TYPE_LENGTH и IL отличные от 0 (f10). На вход подается последовательность сообщений, передающих части заданной полезной нагрузки. Конечное и промежуточные сообщения имеют корректные значения полей TYPE_LENGTH и IL, равные 0. Выходом функции является некорректная последовательность сообщений, в которой как у конечного, так и у промежуточных сообщений значения полей TYPE_LENGTH и IL отличны от 0 (изменения вносятся отдельно либо в последнее сообщение, либо в промежуточные).

Функция формирования сообщений с фрагментированной полезной нагрузкой, в которых последнее и промежуточные сообщения в поле TNF не содержат значение 0x06 (f11). На вход подается последовательность сообщений, передающих части заданной полезной нагрузки, у которых корректно расставлены значения поля TNF. Выходом функции является некорректная последовательность сообщений, в которой у последнего и промежуточных сообщений значения поля TNF отличны от 0x06 (изменения вносятся отдельно либо в последнее сообщение, либо в промежуточные).

Функция формирования сообщений с фрагментированной полезной нагрузкой с установкой флага CF в последнем сообщении (f12). На вход подается последовательность сообщений с корректно расставленными флагами CF, передающих части заданной полезной нагрузки. Выходом функции является некорректная последовательность сообщений, в которой у последнего сообщения установлен флаг CF.

Функция формирования сообщений с фрагментированной полезной нагрузкой, инкапсулированной в несколько NDEF сообщений (f13). На вход подается последовательность сообщений, передающих части заданной полезной нагрузки. Выходом функции является некорректная последовательность сообщений, содержащая несколько отдельных сообщений NDEF, по которым была распределена исходная полезная нагрузка последовательности сообщений, переданной на вход (исходная полезная нагрузка распределяется между несколькими отдельными несвязанными NDEF сообщениями).

Функция формирования сообщений с фрагментированной полезной нагрузкой с флагом ME в первом сообщении (f14). На вход подается последовательность сообщений, передающих части заданной полезной нагрузки. Выходом функции является некорректная последовательность сообщений, в которой у первого сообщения установлен флаг ME.

Функция формирования сообщений с фрагментированной полезной нагрузкой с установкой флага ME в промежуточные сообщения (f15). На вход подается последовательность сообщений с корректно расставленными флагами ME, передающих части заданной полезной нагрузки. Выходом функции является некорректная последовательность сообщений, в которой у нескольких промежуточных сообщений выставлен флаг ME.

Создание конфликтов в сообщениях типа «Text Record»

Функция формирования сообщения типа «TextRecord» с некорректным значением поля PAYLOAD_LENGTH (f16). На вход подается корректное сообщение типа «TextRecord». Выходом функции является некорректное сообщение, в котором значение поля PAYLOAD_LENGTH не совпадает с реальной длиной поля PAYLOAD.

Функция формирования сообщения типа «TextRecord», в котором сообщение закодировано отличным от указанного в поле «Statusbyte» методом (f17). На вход подается корректное сообщение типа «TextRecord». Выходом функции является некорректное сообщение, в поле «Statusbyte» которого была изменена кодировка (также можно изменить кодировку полезной нагрузки, оставляя нетронутым поле «Statusbyte»).

Функция формирования сообщения типа «TextRecord», в котором длина названия кода языка отлична от указанной в поле «Statusbyte» (f18). На вход подается корректное сообщение типа «TextRecord». Выходом функции является некорректное сообщение, в поле «Statusbyte» которого была изменена длина кода языка.

Функция формирования сообщения типа «TextRecord», в котором сообщение написано на языке, отличном от указанного в поле «Statusbyte» (f19). На вход подается корректное сообщение типа «TextRecord». Выходом функции является некорректное сообщение, в котором содержится новый текст на языке, отличном от указанного в поле «Statusbyte».

Создание конфликтов в сообщениях типа URI

Функция формирования сообщения типа URI с некорректным кодом протокола (f20). На вход подается корректное сообщение типа URI. Выходом функции является некорректное сообщение, в котором поле «Identifiercode» содержит число, превышающее 35.

Функция формирования сообщения типа URI с некорректным URI (f21). На вход подается корректное сообщение типа URI. Выходом функции является сообщение, в котором указан некорректный адрес URI (некорректный или недоступный).

Создание конфликтов в сообщениях типа «Smart Poster»

Функция формирования сообщения типа «SmartPoster», в котором отсутствует один из обязательных типов записей (f_{22}). На вход подается корректное сообщение типа «SmartPoster». Выходом функции является некорректное сообщение типа «SmartPoster», в котором отсутствует запись типа URI.

Функция формирования сообщения типа «SmartPoster», в котором присутствует несколько записей типа URI (f_{23}). На вход подаются два сообщения типа URI. Выходом функции является некорректное сообщение типа «SmartPoster», в котором присутствует две записи типа URI.

Функция формирования сообщения типа «SmartPoster», в котором присутствует несколько записей «TitleRecord» с одинаковыми языковыми идентификаторами (f_{24}). На вход подается два сообщения типа Text, содержащие одинаковые языковые идентификаторы. Выходом функции является некорректное сообщение типа «SmartPoster», в котором присутствует две записи «TitleRecord» с одинаковыми языковыми идентификаторами.

Функция формирования сообщения типа «SmartPoster», в котором присутствует некорректная запись типа Action (f_{25}). На вход подается корректное сообщение типа «SmartPoster» с записью типа Action. Выходом функции является некорректное сообщение типа «SmartPoster», в котором значение операции в записи Action лежит вне промежутка чисел от 0 до 2.

Функция формирования сообщения типа «SmartPoster», в котором присутствует запись типа Size, содержащая некорректное значение (f_{26}). На вход подается корректное сообщение типа «SmartPoster» с записью типа Size. Выходом функции является некорректное сообщение типа «SmartPoster», в котором значение Size не соответствует размеру объекта, на который ссылается URI.

Функция формирования сообщения типа «SmartPoster», в котором присутствует запись типа Type, содержащая некорректное значение (f_{27}). На вход подается корректное сообщение типа «SmartPoster» с записью типа Type. Выходом функции является некорректное сообщение типа «SmartPoster», в котором значение Type не соответствует типу объекта, на который ссылается URI.

Описание разработанного метода генерации

Пусть $F = \{f_1, \dots, f_{27}\}$– множество разработанных функций генерации некорректных сообщений NDEF.Данные функции формируют некорректные пакеты или

последовательности пакетовNDEF на основе заранее подготовленных корректных сообщений. Корректные сообщения могут быть получены, к примеру, с помощью сниффера NFC или из базы заранее подготовленных сообщений. Пусть $Collect(t) = G$ – функция получения корректных сообщений NDEF, где t– численный параметр, влияющий на количество собираемых пакетов (к примеру, время сбора в случае использования сниффера), P– множество перехваченных пакетов NDEF. Опишем алгоритм генерации некорректных данных для проведения фаззинг-тестирования.

1. Вход: множество функций генерации F, количество итераций алгоритма *Iters*, численный параметр для сбора корректных сообщенийT.

2. Выход: *Result* – множество последовательностей сообщений NDEF для проведения фаззинг-тестирования.

3. Описание шагов:

3.1. Положить $funcs \leftarrow F$, $iters \leftarrow Iters$, $t \leftarrow T$.

3.2. Положить $i \leftarrow 0$.

3.3. Положить $M \leftarrow \{\}$.

3.4. Пока $i < iters$ выполнить:

3.4.1. Собрать корректные пакеты $packs \leftarrow Collect(t)$.

3.4.2. Для каждого пакета p в $packs$ выполнить:

3.4.2.1. Для каждой функции f в $funcs$ выполнить:

3.4.2.1.1. Положить $ep \leftarrow f(p)$.

3.4.2.1.2. Положить $M \leftarrow M \cup \{ep\}$.

3.4.3. Положить $i \leftarrow i + 1$.

3.5. Положить $Result \leftarrow M$.

3.6. Вернуть $Result$.

В результате $Result$ будет содержать множество последовательностей ep с некорректными пакетами NDEF для проведения фаззинг-тестирования приложений NFC. При этом функции из множества F должны выполнять проверку корректности подаваемых на вход данных.

Заключение

В настоящее время технология NFC используется множеством различных устройств и приложений, что, помимо появления новых возможностей в области обмена данными, также способствует росту атак, проводимых с помощью эксплуатации уязвимостей. Данные уязвимости могут возникать из-за ошибок в реализации модулей NFC. Для поиска подобных ошибок в программном обеспечении может использоваться фаззинг-тестирование, при этом для проведения фаззинга необходимо генерировать множество некорректных данных.

В связи с этим в рамках настоящей работы разработан метод генерации некорректных данных для проведения фаззинг-тестирования на основе внесения конфликтов в протокол сообщений NDEF, используемый при передаче данных с помощью NFC. Приведено описание функций, создающих различные конфликты в сообщениях NDEF, а также описание алгоритма генерации последовательностей некорректных сообщений.

Дальнейшие исследования в области фаззинг-тестирования NFC могут быть направлены на разработку интеллектуального метода оценки покрытия тестируемого приложения, что позволит внести в разработанный метод модификации для повышения эффективности фаззинг-тестирования.

Литература

1. Share of smartphones with near-field communication (NFC) chips marketed in Russia from 2019 to 2022. [Электронный ресурс].URL: https://www.statista.com/statistics/1308533/russia-share-of-smartphones-with-near-field-communication-chips/ (Дата обращения: 04.12.2023).

2. CVE-2023-35671 Detail.[Электронный ресурс]. URL: https://nvd.nist.gov/vuln/detail/CVE-2023-35671 (Дата обращения: 04.12.2023).

3. *Томилов И. О., Трифанов А. В.* Фаззинг. Поиск уязвимостей в программном обеспечении без наличия исходного кода //Интерэкспо Гео-Сибирь. – 2017. – Т. 9. – №. 2. – С. 75-80.

4. *Wang Z.* et al. GNFCVulFinder: NDEF Vulnerability Discovering for NFC-Enabled Smart Mobile Devices Based on Fuzzing //Security and Communication Networks. – 2021. – Т. 2021. – С. 1-14.

5. *Miller C.* Exploring the NFC attack surface //Proceedings of Blackhat. – 2012.

6. Связь ближнего радиуса действия (NFC). [Электронный ресурс]. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/protocols/nfc/index.html (Дата обращения: 04.12.2023).

7. NFC Data Exchange Format (NDEF). Technical Specification. NFC ForumTM. NDEF 1.0. NFCForum-TS-NDEF_1.0– Введ. 2006-07-24.

8. RFC 2046. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types– Введ. ноябрь 1996.

9. RFC 3986. Uniform Resource Identifier (URI): Generic Syntax – Введ.январь 2005.

10. Text Record Type Definition. Technical Specification. NFC ForumTM. RTD-Text 1.0. NFCForum-TS-RTD_Text_1.0 – Введ.2013-08-14.

11. URI Record Type Definition. Technical Specification. NFC ForumTM. RTD-URI 1.0. NFCForum-TS-RTD_URI_1.0 – Введ. 2006-07-24.

12. Smart Poster Record Type Definition. Technical Specification. NFC ForumTM. SPR 1.1. NFCForum-SmartPoster_RTD_1.0 – Введ. 2006-07-24.

13. *Гаврилюк В.И.* Протоколы POP, IMAP, SMTP: основные принципы и применение // Молодой ученый. 2020. № 19(309). С. 119–121.

14. *Клюева Е.Г., Шорин А.Н.* Анализ безопасности почтовых серверов, функционирующих под управлением операционных систем Linux/Windows // Информационные технологии в образовании и науке: Сб. материалов науч.-практич. межвузов. интернет-конференции с международным участием, посвящ. 70-летию Атырауского гос. ун-та им. Халела Досмухамедова, Атырау (8 октября 2019 г.). Атырау: Атырауский гос. ун-т им. Халела Досмухамедова. 2019. С. 156–159.

15. *Лобов И.Ж.* Анализ устойчивости почтового сервера Microsoft exchange к сетевым угрозам на основе результатов моделирования // Вопросы устойчивого развития общества. 2022. № 9. С. 334–343.

16. *Головина Е.Ю., Фархутдинова А.И.* Разработка программного средства проверки электронных писем для почтового сервера предприятия // Информационные технологии. Проблемы и решения. 2022. № 2(19). С. 62–67.

17. *Чернышов М.К.* Использование механизма Split DNS при развертывании корпоративного почтового сервера // Информатика: проблемы, методы, технологии: Материалы XX Междунар. науч.-методич. конф. (г. Воронеж, 13–14 февраля 2020 г.) / Под ред. *А.А. Зацаринного, Д.Н. Борисова*. Воронеж: ООО «Вэлборн». 2020. С. 210–214.

18. *Mahmoud A.B., Grigoriou N., Fuxman L., et al.* Email is evil ! Behavioural responses towards permission-based direct email marketing and gender differences // Journal of Research in Interactive Marketing. 2019. V. 13. № 2. P. 227–248. DOI: 10.1108/JRIM-09-2018-0112.

19. *Sobotta N.* Why forwarded email threads are hard to read: The Email format as an antecedent of Email overload // Communications of the Association for Information Systems. 2016. V. 39. № 1. P. 16–31. DOI: 10.17705/1cais.03902.

20. *Gan S., Qin X., Tu X., et al*. Path Sensitive Fuzzing for Native Applications // IEEE Transactions on Dependable and Secure Computing. 2022. V. 19. № 3. P. 1544–1561. DOI: 10.1109/TDSC.2020.3027690.

21. *Zakeri Nasrabadi M., Parsa S., Kalaee A*. Format-aware learn&fuzz: deep test data generation for efficient fuzzing // Neural Computing & Applications. 2021. V. 33. № 5. P. 1497–1513. DOI: 10.1007/s00521-020-05039-7.

22. *Зимин Е.Е.* Методика фаззинг-тестирования кода с помощью AFL // Безопасные информационные технологии: Сб. трудов XI Междунар. науч.-технич. конф. (Москва, 6–7 апреля 2021 г.). М.: МГТУ имени Н.Э. Баумана (НИУ). 2021. С. 124–129.

23. *Тронов К.А., Белов Ю.С.* Оптимизация инструментария afl для лучшего покрытия кода при работе со специфичными данными // E-Scio. 2021. № 5(56). С. 566–571.

24. *Алексеев Д.М., Иваненко К.Н., Убирайло В.Н.* Fuzzing как метод тестирования программ: преимущества и недостатки // Наука в современном обществе: закономерности и тенденции развития: Сб. статей междунар. науч.-практич. конф. (г. Пермь, 25 февраля 2017 г.) В 2-х частях. Ч. 2. Пермь: ООО «Аэтерна». 2017. С. 18–19.

25. *Язов Ю.К., Кадыков В.Б., Енютин А.Ю., Суховерхов А.С.* Использование технологии фаззинга для поиска уязвимостей в программно-аппаратных средствах автоматизированных систем управления технологическими процессами // Программная инженерия. 2011. № 6. С. 44–47.

DEVELOPMENT OF THE METHOD FOR GENERATING INCORRECT DATA FOR FUZZ TESTING OF APPLICATIONS USING NFC

NIKOLAY.N. SAMARIN
Ph D., Moscow, Russia, samarin_nik@mail.ru

ABSTRACT

Introduction: This article proposes a method for generating incorrect data to conduct fuzz testing of applications using NFC. The NDEF message protocol used in data transmission via NFC was examined during the research. Based on the protocol analysis, possible conflicts and contradictions that can be introduced into NDEF messages were identified. The functions that create various conflicts in NDEF messages and the algorithm for generating sequences of incorrect messages using these functions are described. Further research is planned to develop an intelligent method for evaluating the coverage of NFC-tested applications, which will allow modifications to be made to the developed method to improve the effectiveness of fuzz testing.

Keywords: information security; fuzzing; NFC; NDEF.

REFERENCES

1. Share of smartphones with short-range communication (NFC) chips sold in Russia in the period from 2019 to 2022. [electronic resource]. URL: https://www.statista.com/statistics/1308533/russia-share-of-smartphones-with-near-field-communication-chips / (Date of reference: 04.12.2023).
2. Detail CVE-2023-35671. [Electronic resource]. URL: https://nvd.nist.gov/vuln/detail/CVE-2023-35671 (Date of reference: 04.12.2023).
3. Tomilov I. O., Trifanov A.V. Fuzzing. Finding vulnerabilities in software without the source code. Interexpo Geo-Siberia. 2017. Vol. 9. No. 2. Pp. 75-80.
4. Wang Z. and others. GNFCVulFinder: NDEF vulnerability detection for NFC-enabled intelligent mobile devices based on fuzzing. Security and Communication Networks. 2021. T. 2021. Pp. 1-14.
5. Miller S. Investigation of the surface of NFC attacks. Proceedings of Blackhat. 2012.
6. Communication with the state (NFC). [electronic resource]. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/protocols/nfc/index.html (Date of access: 04.12.2023).
7. NFC data exchange format (NDEF). Technical specifications. NFC ForumTM. NDEF 1.0. NFCForum-TS-NDEF_1.0– November 2006-07-24.
8. RFC 2046. Multifunctional Internet Mail Extensions (MIME), Part Two: Media Types Introduction. November 1996.
9. RFC 3986. Single Resource Identifier (URI): Common syntax Early January 2005.
10. Determining the type of text entry. Technical specification. NFC ForumTM. RTD-Text 1.0. NFCForum-TS-RTD_Text_1.0 – November 2013-08-14.
11. Determining the type of URI entry. Technical specifications. NFC ForumTM. RTD-URI 1.0. NFCForum-TS-RTD_URI_1.0 – November 2006-07-24.
12. Determining the type of record for an intelligent poster. Technical specifications. NFC ForumTM Forum. SPR 1.1. NFC Forum-SmartPoster_RTD_1.0 – November 2006-07-24.
13. Gavriljuk V.I. Protokoly POP, IMAP, SMTP: osnovnye principy i primenenie. Molodoj uchenyj. 2020. № 19(309). Pp. 119–121 (in Russian).
14. Kljueva E.G., Shorin A.N. Analiz bezopasnosti pochtovyh serverov, funkcionirujushhih pod upravleniem operacionnyh sistem Linux/Windows. Informacionnye tehnologii v obrazovanii i nauke: Sb. materialov nauch.-praktich. mezhvuzov. internet-konferencii s mezhdunarodnym uchastiem, posvjashh. 70-letiju Atyrauskogo gos. un-ta im. Halela Dosmuhamedova, Atyrau (8 oktjabrja 2019 g.). Atyrau: Atyrauskij gos. un-t im. Halela Dosmuhamedova. 2019. Pp. 156–159 (in Russian).
15. Lobov I.Zh. Analiz ustojchivosti pochtovogo servera Microsoft exchange k setevym ugrozam na osnove rezul'tatov modelirovanija. Voprosy ustojchivogo razvitija obshhestva. 2022. № 9. Pp. 334–343 (in Russian).
16. Golovina E.Ju., Farhutdinova A.I. Razrabotka programmnogo sredstva proverki jelektronnyh pisem dlja pochtovogo servera predprijatija. Informacionnye tehnologii. Problemy i reshenija. 2022. № 2(19). Pp. 62–67 (in Russian).
17. Chernyshov M.K. Ispol'zovanie mehanizma Split DNS pri razvertyvanii korporativnogo pochtovogo servera. Informatika: problemy, metody, tehnologii: Materialy XX Mezhdunar. nauch.-metodich. konf. (g. Voronezh, 13–14 fevralja 2020 g.). Pod red. A.A. Zacarinnogo, D.N. Borisova. Voronezh: OOO «Vjelborn». 2020. Pp. 210–214 (in Russian).
18. Mahmoud A.B., Grigoriou N., Fuxman L., et al. Email is evil ! Behavioural responses towards permission-based direct email marketing and gender differences. Journal of Research in Interactive Marketing. 2019. Vol. 13. № 2. Pp. 227–248. DOI: 10.1108/JRIM-09-2018-0112.
19. Sobotta N. Why forwarded email threads are hard to read: The Email format as an antecedent of Email overload. Communications of the Association for Information Systems. 2016. Vol. 39. № 1. Pp. 16–31. DOI: 10.17705/1cais.03902.
20. Gan S., Qin X., Tu X., et al. Path Sensitive Fuzzing for Native Applications. IEEE Transactions on Dependable and Secure Computing. 2022. Vol. 19. № 3. Pp. 1544–1561. DOI: 10.1109/TDSC.2020.3027690.

21. Zakeri Nasrabadi M., Parsa S., Kalaee A. Format-aware learn&fuzz: deep test data generation for efficient fuzzing. Neural Computing & Applications. 2021. Vol. 33. № 5. Pp. 1497–1513. DOI: 10.1007/s00521-020-05039-7.
22. Zimin E.E. Metodika fazzing-testirovanija koda s pomoshh'ju AFL. Bezopasnye informacionnye tehnologii: Sb. trudov XI Mezhdunar. nauch.-tehnich. konf. (Moskva, 6–7 aprelja 2021 g.). M.: MGTU imeni N.Je. Baumana (NIU). 2021. Pp. 124–129 (in Russian).
23. Tronov K.A., Belov Ju.S. Optimizacija instrumentarija afl dlja luchshego pokrytija koda pri rabote so specifichnymi dannymi. E-Scio. 2021. № 5(56). Pp. 566–571 (in Russian).
24. Alekseev D.M., Ivanenko K.N., Ubirajlo V.N. Fuzzing kak metod testirovanija programm: preimushhestva i nedostatki. Nauka v sovremennom obshhestve: zakonomernosti i tendencii razvitija: Sb. statej mezhdunar. nauch.-praktich. konf. (g. Perm', 25 fevralja 2017 g.) V 2-h chastjah. Ch. 2. Perm': OOO «Ajeterna». 2017. Pp. 18–19 (in Russian).
25. Jazov Ju.K., Kadykov V.B., Enjutin A.Ju., Suhovephov A.S. Ispol'zovanie tehnologii fazzinga dlja poiska ujazvimostej v programmno-apparatnyh sredstvah avtomatizirovannyh sistem upravlenija tehnologicheskimi processami. Programmnaja inzhenerija. 2011. № 6. Pp. 44–47 (in Russian).