УДК № 004.423.4:519.686.2

А.И. Гетьман, В.А. Падарян

г. Москва, Институт системного программирования РАН.

ВОССТАНОВЛЕНИЕ ФОРМАТА ДАННЫХ ПУТЕМ АНАЛИЗА БИНАРНОГО КОДА: СОСТОЯНИЕ И ПЕРСПЕКТИВЫ*

В статье рассматриваются вопросы применения динамического анализа бинарного кода для решения задачи восстановления форматов данных. Описываются аспекты затрудняющие анализ, такие как шифрование данных и способы их преодоления. Перечисляются области применения восстановленных форматов.

Ключевые слова: динамический анализ; бинарные трассы; восстановление форматов; анализ протоколов

A.I. Getman, V.A. Padaryan

DATA FORMAT RECOVERY THROUGH BINARY CODE ANALYSIS: STATE AND PERSPECTIVES

In this paper we discuss applying dynamic binary code analysis approach to data format recovery problem. We describe aspects that complicate the analysis such as data encryption and propose ways of mitigating them. We also enumerate the areas where recovered data formats can be used.

Keywords: dynamic analysis; binary traces; format recovery; protocol analysis

Введение

Анализ бинарного кода, в первую очередь, связан с выделением и анализом реализованных в программах алгоритмов, с целью их понимания и формального описания [1]. Для решения этих задач одним из промежуточных шагов, как правило, является повышение уровня представления программы [2]. Реализация алгоритма, согласно определению Вирта, представляет собой совокупность кода и данных, обрабатываемых этим кодом. Внутри программы на языке высокого уровня данные хранятся и передаются в виде переменных, имеющих некоторые типы в системе типов языка программирования, на котором написана программа. В процессе компиляции операции с переменными этих типов преобразуются в конструкции на языке ассемблера соответствующей процессорной архитектуры. Задача восстановления типов, то есть получение высокоуровневых типов данных отдельных переменных по ассемблерной программе, является частью задачи декомпиляции, равно как и идентификация самих переменных. Подходы к решению этих задач зависят как от процессорной архитектуры, так и от особенностей компилятора. С другой стороны, в

* Работа поддержана грантом РФФИ 14-07-00606

ходе обмена данными с другими системами программа может воспользоваться либо одним из видов внешней памяти (например, жёстким диском), либо каналом связи, таким как сеть Ethernet. При этом данные принимают форму файлов и сетевых пакетов, формат которых должен поддерживаться другими системами и, таким образом, в общем случае не связан с особенностями конкретной архитектуры и компилятора. Например, поддержка одних и тех же протоколов стека TCP/IP реализуется в самых разных ОС и архитектурах, а формат исполняемых файлов PE может применяться в процессорных архитектурах x86, MIPS, ARM, PowerPC и многих других. Таким образом, формат данных не привязан ни к конкретной аппаратно-программной платформе, ни к средствам получения исполняемого кода программ, которые этот формат обрабатывают. Наличие формата данных, обрабатываемых конкретной реализацией алгоритма позволит формализовать его описание, не привязанное к конкретной программно-аппаратной платформе.

В настоящий момент, восстановление форматов выполняется в основном в ручном режиме, что требует высокой квалификации и является очень затратной по времени задачи. В частности, восстановление протокола CIFS/SMB, в рамках реализации сервера Samba, заняло более 10 лет. Поэтому задача автоматизации этого процесса является весьма актуальной.

Далее в разделе 1 будут рассмотрены области применения восстановленных форматов. Процесс восстановления формата состоит из трёх основных частей: выделение границ минимальных информационных единиц сообщения – полей, группировка полей в иерархическую структуру и восстановление семантики отдельных полей, таких как поля-разделители, задающих размер полей переменной длины. Более подробно понятие формата данных рассматривается в разделе 2. В разделе 3 приведено описание алгоритма восстановления формата с использованием динамического анализа бинарного кода. Раздел 4 содержит описание более общей задачи восстановления протокола и подходов к её решению.

1. Область применения описаний форматов данных

Анализ алгоритмов является важной, но далеко не единственной задачей, в которой требуется описание форматов данных. Среди таких задач, в первую очередь, можно назвать поиск уязвимостей с помощью систем автоматизированного тестирования программ на основе фаззинга. К таким системам относятся, например, Packet Vaccine, ShieldGen и Peach. Одним из ключевых аспектов фаззинга является генерация входных данных, которые являются синтаксически корректными, то есть принимаются программой, но приводят её в некорректное состояние. От того, насколько хорошо система фаззинга подбирает входные данные, во многом зависит как скорость тестирования, так и его успешность. Наличие описания формата входных данных способно значительно улучшить указанные характеристики тестирования.

Другой важной задачей из сферы информационной безопасности является предотвращение несанкционированного доступа по сети. Для решения этой задачи используются системы обнаружения и предотвращения вторжений (IPS и IDS), такие как Snort и Bro. В основе этих систем лежат алгоритмы классификации трафика, позволяющие привязать сетевые пакеты к некоторому протоколу и приложению, которое его использует. В процессе классификации выполняется разбор сетевых пакетов, с использованием технологии DPI. Для применения этой технологии требуются подпрограммы-разборщики сетевых пакетов, которые могут автоматически

генерироваться по восстановленным форматам. Аналогичные программы-разборщики применяются и в других, менее специализированных системах анализа сетевого трафика, таких как Wireshark.

Ещё один важный класс задач, для решения которых знание форматов сетевых сообщений критически важно – анализ вирусного ПО. В настоящее время, достаточно широко распространено вирусное ПО, которое позволяет формировать из заражённых компьютеров крупномасштабные сети – ботнеты. Среди вирусов, обладающих таким функционалом можно выделить MegaD и agobot. Управление сетью из заражённых машин осуществляется с помощью специальных командных протоколов, знание которых позволяет, с одной стороны, оценить функционал конкретного ботнета, а с другой, способствовать выявлению заражённых машины и дезактивации вируса.

Ещё одной задачей, решение которой восстановление формата может существенно облегчить – сравнительное тестирование нескольких реализаций одного протокола. Различия в форматах, восстановленным по двум реализациям одного и того же протокола могут указывать на ошибку в одной из реализаций.

Из приведённых примеров можно заключить, что есть два основных варианта использования описания формата данных: разбор соответствующих формату данных, и генерация синтаксически корректных данных.

2. Понятие формата данных и автомата протокола

Множество сетевых пакетов или файлов, соответствующих некоторому формату образуют язык, а сами пакеты или файлы являются корректными словами этого языка. Одной из удобных математических форм представления этого языка могут служить атрибутные грамматики, в которой правила вывода описывают структуру данных, а в атрибутах содержится семантическая информация, позволяющая значения этих данных интерпретировать.

Следует отметить, что в различных работах, посвящённых вопросам автоматического восстановления форматов, как форма представления формата данных, так и набор атрибутов может существенно отличаться, в зависимости от возможностей конкретных алгоритмы автоматического восстановления. На рис.1 приведён пример работы программы, осуществляющей обработку данных в некотором формате.

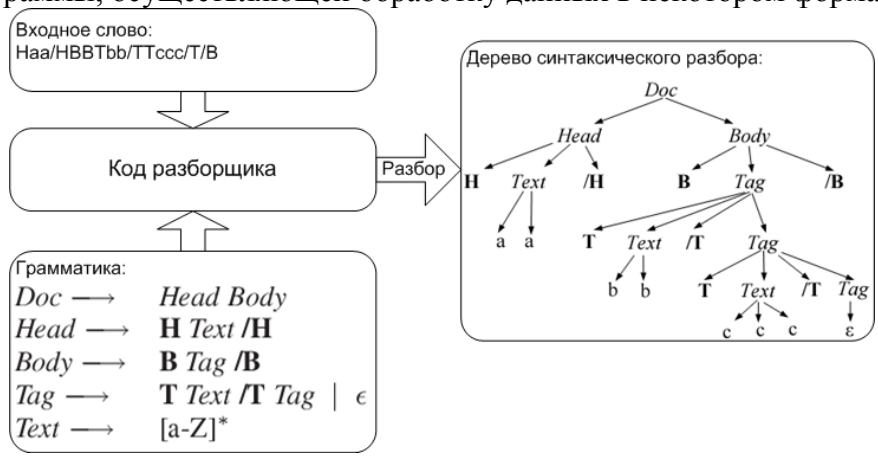

Рис. 1. Пример работы программы при обработке данных.

На данном примере можно проиллюстрировать основные понятия, использующиеся при описании структуры сообщения. Минимальная структурная единица формата – поле. В данном примере можно выделить поля двух типов:

- поля, содержащие отдельные буквы алфавита,
- поля, содержащие специальные символы `H`, `B`, `T`, `/H`, `/B`, `/T`, ϵ.

В более сложных бинарных протоколах отдельные поля содержат числа, как с фиксированной, так и с плавающей точкой могут иметь размер от 1 до 16 байт.

Как видно из грамматики, лексема `Text` может состоять из переменного числа символов. Для обозначения таких структурных единиц используют термин *последовательность*. Длина последовательности может задаваться различными способами, наиболее распространёнными из которых являются поля разделители, содержащие специальное терминальное значение (ϵ в примере) и поля длины, содержащие число, задающее размер последовательности.

Ещё одним видом группировки является *запись*, в отличие от последовательности, оно содержит фиксированное число элементов. Примером записи из двух элементов на рисунке является лексема `Doc`, состоящая из элементов `Head` и `Body`.

В приведённом примере есть несколько видов полей, обладающих специальной семантикой. Первая группа – поля разделители, обозначающие конец некоторой лексемы. В примере к таким полям относятся символы ϵ, `/H`, `/B`, `/T`. Поля со специальными значениями `H`, `B`, `T` называются ключевыми полями, поскольку содержат специальные константные значения, задаваемые протоколом. В рассматриваемом примере они используются для задания начала лексем, но в общем случае могут использоваться для самых разных задач.

Передача данных по сети осуществляется с помощью пересылки сообщений между участниками обмена. Целью обмена обычно является получение некоторой информации одной стороной обмена (клиентом) у другой стороны (сервера). В более сложных случаях обе стороны обмена могут играть роль и сервера и клиента.

Можно выделить две типичные схемы обмена: без установления соединения и с установлением соединения. В первом случае обмен данными может быть представлен как последовательность несвязанных пар сообщений вида «запрос»-«ответ». В этой схеме, единственным условием успешного обмена данными является корректность сообщений, соответствующих запросу и ответу – они оба должны являться словами соответствующего языка.

Во втором случае, вначале осуществляется предварительный обмен сообщениями для установления соединения, затем участники обмениваются сообщениями с данными, после чего происходит обмен сообщениями, сигнализирующими о разрыве соединения. Для реализации различных действий, таких как установление и разрыв соединения, а также передачи данных используются сообщения разного типа, то есть эти сообщения могут соответствовать словам разных языков. В случае схемы с установлением соединения, используется понятие *сессия* – последовательность сообщений, которые передаются между участниками обмена в рамках некоторого протокола, от момента установления соединения до момента его разрыва. В этом случае условия успешного обмена выглядят более сложно. Необходимым, но недостаточным условием является то, что все сообщения должны быть словами соответствующих языков. Для корректного обмена, сообщения должны идти в правильном порядке. Например, сообщение о завершении соединения не должно приходить раньше последнего сообщения с данными. То есть сессия должна быть

представлена корректной последовательностью сообщений. Для описания корректных последовательностей используется автомат состояний, который может быть наглядно представлен в виде графа. Пример автомата состояний для протокола TCP приведён на рис. 2. Вершины графа соответствуют состоянию приложения, когда оно ожидает сообщение, относящееся к некоторому множеству типов. Каждому типу сообщения соответствует ребро, переводящее приложение (и автомат) в новое состояние. Сетевой протокол представляет собой формальную спецификацию правил обмена. Спецификация содержит, с одной стороны формальное описание языков всех типов сообщений, входящих в протокол, а с другой – описание автомата состояний протокола.

Важной задачей обратной инженерии является задача восстановления протокола, для решения которой требуется:

- восстановить форматы всех сообщений, входящих в протокол
- восстановить автомат протокола

Более подробная информация об определении формата данных, видах семантики и подходах к восстановлению форматов содержится в работах [3-5].

Процесс считывания файла с устройства хранения данных также можно рассматривать как процесс обмена данными между двумя участниками. Одним из участников обмена является файловая система, а другим - приложение, запрашивающее данные файла. Файл в этом случае может рассматриваться как единое целое, а понятие сессия используется для обозначения последовательности операций с этим файлом. Далее для обозначения объекта, формат которого восстанавливается, будет использоваться общий термин *сообщение*, обозначающий сетевое сообщение или файл.

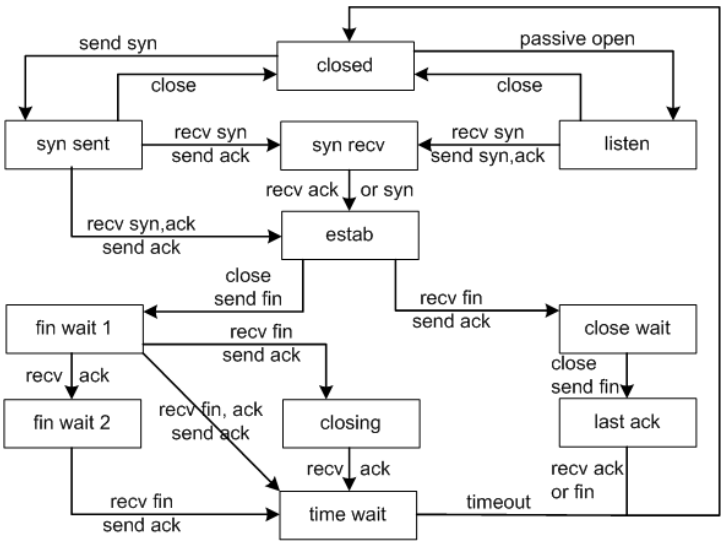

Рис. 2. Представление автомата состояния протокола TCP в виде графа

3. Алгоритм восстановления форматов данных

На данный момент наиболее перспективным подходом к восстановлению форматов данных является подход с использованием динамического анализа кода. Косвенно перспективность данного подхода подтверждается количеством научных работ, описывающих различные реализации данного подхода. В отличие от статистического анализа сетевого трафика, динамический анализ позволяет восстановить большое количество семантической информации, благодаря наблюдению

за процессом обработки сетевого сообщения. Преимуществом перед статическим анализом кода является более высокая точность границ отдельных полей и других структурных элементов, так как адреса всех обращений в память известны и отсутствует проблема алиасинга. Более подробно сравнение различных подходов разобрано в работе [3].

Из приведённого на рис. 1 примера видно, что в процессе разбора сообщения, данные, подаваемые на вход в виде некоторого буфера фиксированного размера, преобразуются в форму дерева синтаксического разбора. Это преобразование необязательно явно выражено в коде программы, однако путём анализа процесса разбора сообщения, находящегося в известном буфере можно с достаточно высокой точностью восстановить синтаксическую структуру сообщения. На основе этой структуры, путём её обобщения можно получить описание языка в виде частичной грамматики, которую затем можно дополнить на основании анализа разбора других сообщений этого же формата.

Для получения информации о процессе разбора сообщения применяется метод снятия трассы в процессе обработки сообщения, при котором программа, осуществляющая обработку, запускается внутри эмулятора. Альтернативным способом получения трассы может быть метод на основе динамического или статического инструментирования с использованием таких средств как Valgrind, Pin или DynamoRIO, однако эти методы являются инвазивными по отношению к программной системе и могут быть обнаружены программой, если она применяет некоторый вариант защиты от анализа. Общая схема метода приведена на рис. 3.

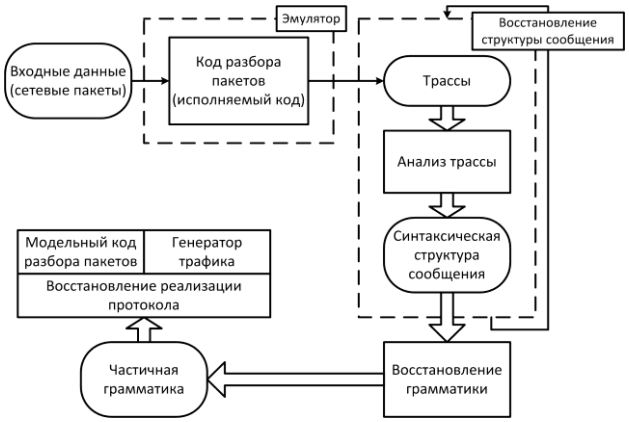

Рис. 3. Общая схема восстановления формата на основе динамического подхода

Так как программа может обрабатывать только данные, которые лежат в оперативной памяти и на регистрах, то в процессе обработки любому объекту, такому как файл или сетевое сообщение, соответствует буфер в оперативной памяти, в котором в некоторый момент времени находятся данные этого объекта. Таким образом, первой задачей анализа является определение границ этого буфера и временного диапазона, когда он содержит данные сообщения. Для этого требуется определять моменты вызовов функций получения данных объекта (чтение файла, получение сетевого пакета) и анализировать их фактические параметры. В случае инвазивного подхода для этого используются обратные вызовы, а в случае неинвазивного – модели функций ввода/вывода, описанные, например, в работе [3]. Следует отметить, что в общем

случае, может не существовать единого буфера, содержащего весь объект в один момент времени. Сложности, возникающие в таких ситуациях, будут описаны ниже.

После того, как буфер определён, алгоритм восстановления выполняет следующие шаги:

1. Выделение алгоритма обработки буфера
2. Восстановление синтаксической структуры сообщения:
   - Восстановление границ отдельных полей сообщения, путём анализа размеров отдельных инструкций доступа к данным буфера
   - Иерархическая группировка полей на основе структурного анализа графа потока управления
3. Восстановление семантической информации

Схема алгоритма, приведена на рис. 4. Серым цветом выделены данные, которые должны быть построены по трассе программы до выполнения восстановления.

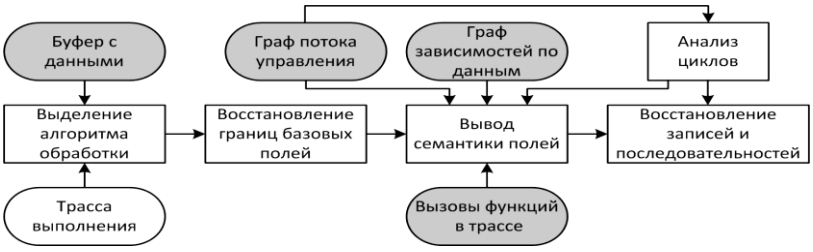

Рис. 4. Схема алгоритма восстановления формата.

Алгоритм выведения семантики основан на алгоритме вывода типов, использующегося в задаче декомпиляции. Вводится понятие источника семантики, то есть точки в трассе программы, в которой семантика использующихся данных точно известна. После получения списка таких точек происходит анализ зависимостей по данным между этими источниками и частями буфера, содержащего сообщение, и делается вывод о семантике этих частей. В качестве источников семантики могут использоваться вызовы библиотечных функций, специфические процессорные инструкции и специфические конструкции в графе потока управления, такие как условные переходы, прерывающие выполнение цикла, по которым можно определять поля длины и разделители.

Основными факторами, влияющими на точность и полноту восстановления формата являются:

- ограничение динамического подхода, приводящего к неполноте получаемой грамматики,
- использование шифрования данных,
- работа программы с несколькими источниками данных (считывание нескольких файлов с разными форматами),
- поступление данных по частям (постепенное считывание файла).

Для дополнения получаемой грамматики используется объединение грамматик, полученных при анализе трасс обработки нескольких сообщений, которое выполняется с помощью модифицированного алгоритма Нидлмана-Вунша, изначально применявшегося в биоинформатике. Алгоритм выполняет сопоставление отдельных элементов двух грамматик и выполняет их слияние.

Для разделения данных получаемых от различных источников в моделях функций ввода/вывода используется значение параметра, содержащего идентификатор источника.

Обработка данных, поступающих по частям, выполняется с помощью построения виртуального буфера, который строится, как склейка из буферов, являющихся параметрами функций ввода/вывода, с учётом вызова функций позиционирования (в случае анализа формата файла).

Для обработки шифрованного трафика, также используется виртуальный буфер, в котором шифрованные участки заменяются на результат расшифрования. Для выполнения анализа, помимо моделей функций ввода/вывода, также требуется наличие моделей функций шифрования/расшифрования. Подходы к автоматизации поиска таких функций и построения их моделей рассмотрены в работе [3].

4. Задача восстановления протокола

Для решения общей задачи восстановления протокола необходима разработка подхода к автоматическому восстановлению автомата протокола. Решение этой задачи подразумевает решение трёх подзадач:

- классификация сообщений протокола по типам, так как именно типам сообщений соответствуют рёбра переходов в графе автомата протокола, пример которого приведён на рис. 2,
- выделение сессий обмена, так как работа автомата протокола происходит в рамках отдельных сессий,
- построение автомата протокола по выделенным сессиям и его обобщение для расширения области применимости.

Существует большое количество подходов к решению задачи классификации на основе различных алгоритмов кластеризации, а также статистического анализа для поиска ключевых полей сообщения, которые определяют его тип. Подходы значительно различаются по точности, а также количеству и типу информации, которую должен предоставить пользователь. В одном из подходов, для вычисления расстояния между кластерами используется длина наибольшей общей подстроки. В подходе с иерархической кластеризацией от пользователя требуется задать условия остановки процесса кластеризации.

Решение задачи выделения сессий зависит от условий, в которых она решается. Если сообщения протокола уже классифицированы, и известно, какие из них отвечают за установление и разрыв соединений, то сессии выделяются как цепочки сообщений начинающиеся с сообщения, отвечающего за установку и заканчивающиеся сообщением, отвечающим за разрыв соединения. Если сообщения не классифицированы, то сессии можно выделить на основе динамического анализа последовательностей вызовов функций открытия/закрытия соединений, при наличии их моделей.

Задача построения и обобщения автомата протокола по известным последовательностям сообщений в рамках отдельных сессий, встретившимся в сетевом трафике, осложняется тем, что необходимо построить, с одной стороны минимальный, а с другой, возможно более полный автомат протокола. Для восстановления автомата протокола применяются подходы на основе интерактивного или автоматизированного вывода грамматики. В рамках этих подходов, типы сообщений рассматриваются как буквы некоторого алфавита, сессии – как слова языка, который может быть описан, как

автоматом, который принимает этот язык, так и некоторой грамматикой. Сложность решения этой задачи связана с рядом факторов. Так, теореме Голда (1967) доказывается, что не существует общего метода вывода даже регулярного языка на основе только слов принадлежащих этому языку (позитивных примеров) – требуются также слова, языку не принадлежащие, называемые негативными примерами. Также в теореме Англуин (1978) показано, что задача построения минимального автомата на основе множеств слов, как принадлежащих, так и не принадлежащих языку слов относится к классу NP-полных задач. Все подходы, на основе вывода грамматики можно разделить на следующие группы.

- Методы на основе внешних данных, например доступности негативных примеров.
- Методы на основе эвристик, использующие некоторые априорные знания о входных данных или синтаксических ограничениях языка.
- Характеристические методы, позволяющие решать задачу вывода грамматики, без дополнительных данных и предположений, если эта грамматика относится к некоторому специальному подмножеству регулярных языков, обходя, таким образом, ограничение теоремы Голда.
- Гибридные методы.

Заключение

Область автоматического анализа протоколов развивается достаточно быстро: получено много практических результатов, в том числе и для протоколов, поддерживающих шифрование. Также об этом говорит появление специализированных сред анализа, таких как проект с открытым исходным кодом NetZob, включающий в себя интегрированные инструменты анализа сообщений и автомата протокола, а также средства для применения полученных результатов для генерации трафика и фаззинга протоколов.

В то же время, заметна тенденция к смещению фокуса внимания от задачи восстановления формата к задаче полного восстановления спецификации протокола. Дальнейший прогресс в данном вопросе, по всей видимости, позволит значительно сократить трудозатраты повысить скорость анализа новых протоколов. Это особенно актуально в сфере противодействия вирусным угрозам.

Библиографический список

1. В.А. Падарян, А.И. Гетьман. Автоматизация динамического анализа бинарного кода. // Материалы конференции РусКрипто’2009. Москва, 2-5 апреля 2009
2. А.И. Аветисян, В.А. Падарян, А.И. Гетьман и др. О некоторых методах повышения уровня представления при анализе защищённого бинарного кода. // Материалы XIX Общероссийской научно-технической конференции Методы и технические средства обеспечения безопасности информации. Санкт-Петербург, 2010, с.97-98
3. А.И. Аветисян, А.И. Гетьман. Восстановление структуры бинарных данных по трассам программ. // Труды Института системного программирования, том 22, 2012, с.95-118.
4. А.И. Гетьман, Ю.В. Маркин, В.А. Падарян и др. Восстановление формата данных. // Труды Института системного программирования, том 19, 2010, с. 195-214.

5. Гетьман А.И., Падарян В.А. Методика восстановления формата данных. // Материалы конференции РусКрипто'2010. Москва, 2-5 апреля 2010.