Список использованных источников
1. Bishop, C. M., 2006. Pattern recognition and machine learning: Springer.
2. Castagna, J.P.; Backus, M.M., 1993. Offset Dependent Reflectivity - Theory and Practice of AVO Analysis. Society of Exploration Geophysicists. ISBN 1-56080-059-3
3. Aminzadeh, F. and de Groot, P., 2006. Neural Networks and Other Soft Computing Techniques with Applications in the Oil Industry, EAGE Publications.
4. Samuel, A., L., 1959. Some studies in machine learning using the game of checkers, in IBM Journal of research and development.
5. Dell'Aversana, P., and M. Vivier, 2009. Expanding the frequency spectrum in Marine CSEM applications: Geophysical Prospecting, Volume 57, Issue 4, pages 573-590.

УДК 004.43, 004.056.57

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ ПАРСИНГА И ВАЛИДАЦИИ БИНАРНЫХ ОБЪЕКТОВ С ПОСЛЕДОВАТЕЛЬНОЙ СТРУКТУРОЙ.

Штейнбрехт Евгений Вадимович
shtjen.kz@mail.ru
Магистрант специальности «6М070400 – Вычислительная техника и программное обеспечение», экспериментальная образовательная программа «Администратор по управлению и защите компьютерных систем и сетей на предприятиях» факультета информационных технологий,
ЕНУ им. Л.Н.Гумилева, Нур-Султан, Казахстан
Научный руководитель – Д.Ж. Сатыбалдина

Валидация файлов согласно официальным спецификациям, выпущенным разработчиками форматов, является важным аспектом обеспечения информационной безопасности. Зачастую вредоносные инъекции происходят в обход официальных стандартов файлов. В связи с этим возможность статически проанализировать файл и выделить в нём расходящиеся с правилами элементы позволяет обнаружить значительное количество вредоносных объектов [1].

В настоящее время активно используются сотни различных форматов, для многих из которых выпущены различные образцы спецификаций. При этом каждый из форматов может иметь десятки версий, отличающихся различными элементами [2]. При этом существующих решений для их валидации существует крайне небольшое количество. Среди них к наиболее известным относятся YARA и Kaitai.

Основное назначение языка YARA – поиск сигнатур внутри файлов. По структуре он находится ближе к классу языков регулярных выражений [3]. Таким образом, программы, написанные на нём, предназначены не для последовательного сканирования и валидации объектов, состоящих из байтовых полей, а для поиска и анализа структур с определёнными характеристиками.

Функционал Kaitai включает в себя парсинг и запись обработанных данных в форматированные структуры. В итоговом результате генерируются объекты, хранящие в себе наборы извлечённых данных [4]. Для непосредственно анализа и валидации бинарных объектов данный язык пригоден относительно слабо.

Таким образом, на данный момент времени отсутствуют продукты для удобного и быстрого создания алгоритмов валидации файлов непосредственно на основе спецификаций. В связи с этим целью исследований, результаты которых представлены в данной работе, является разработка оригинального языка и программного средства для парсинга и валидации бинарных файлов на основе их спецификаций.

На начальном этапе разработки был проанализирован набор файлов различных форматов и их составляющих, которые описаны далее.

Rich Text Format, RTF – проприетарный межплатформенный формат хранения текстовых документов с форматированием, предложенный группами программистов, основавшими компании Microsoft и Adobe, как мета-теговый формат для редактора Word в 1982 году. Текстовый редактор WordPad, встроенный в Microsoft Windows, по умолчанию сохраняет документы в формате RTF [5].

Doc – двоичные файлы формата «doc» содержат больше информации (например, больше информации о форматировании текста, фигуры, сценарии), чем файлы форматов txt, RTF и других, но хуже совместимы с текстовыми процессорами сторонних разработчиков.

MathType MTEF – формат, используемый в Windows и на других платформах для графического описания формул. В частности применяется в офисном текстовом редакторе Microsoft Word.

MS-DTYP – набор общих структур, используемых корпорацией Microsoft в различных форматах и протоколах своей разработки.

OLE – технология связывания и внедрения объектов в другие документы и объекты, разработанная корпорацией Майкрософт. OLE позволяет передавать часть работы от одной программы редактирования к другой и возвращать результаты назад [6].

В результате проведённого анализа была подготовлена спецификация языка, способного работать со всеми вышеперечисленными форматами или их элементами, требующими валидации. Ключевой особенностью построения языка является возможность последовательного и подробного переноса спецификации из текстового описательного формата в исходный код. То есть, программист, использующий данный язык, в процессе переноса стандарта, может идти по разделам и подобъектом, перенося каждый из них независимо, сохраняя их порядок.

В частности, в качестве основного образца был взят стандарт описания форматов компании Microsoft. Итоговый вариант языка включает в себя набор синтаксических конструкций, к которым относятся литералы, поля, ссылки, массивы, проверки, объекты и расширения.

Литералы предназначены для задания константных значений и могут принимать различные форматы, в том числе десятичные, шестнадцатеричные и двоичные числа, строки и символы, что подробнее описано в таблице 1.

Таблица 1. Форматы и описания различных видов литералов

Тип литерала или операции	**Формат**	**Описание**
Десятичное число	%набор цифр%	Стандартное цифровое целочисленное десятичное значение в диапазоне от отрицательной до положительной бесконечности.
Шестнадцатеричное число	0x%набор цифр и символов a-f, A-F%	Стандартное цифровое шестнадцатеричное значение.
Двоичное число	0b%набор нулей и единиц%	Стандартное цифровое двоичное значение.
Сложение, вычитание, умножение, деление	%число 1% %оператор% %число 2%, где %оператор% это символы +, -, *, /, для сложения вычитания, умножения и деления соответственно	Стандартные операции, возвращающие результаты математических операций.
Степень	%число%**%степень числа%	Стандартная операция возведения числа в степень.

Байтовые сдвиги	-*%число% и *-%число% для правого и левого сдвига соответственно	Байтовые циклические сдвиги. Применяются в двух контекстах – проверки значения и его записи.
Битовые операции	%число 1% %оператор% %число 2%, %оператор% это символы *&, *|, *^, *!, для операций И, ИЛИ, НЕ-ИЛИ и НЕ соответственно	Стандартные битовые операции. Применяются в двух контекстах – проверки значения и его записи.
Нуль-терминированая строка	"%набор ASCII символов%"	Строка, заканчивающаяся нулевым символом.
Стандартная строка	'%набор ASCII символов%'	Строка, состоящая только из указанных символов.

Литералы используются для манипуляций с числовыми и символьными значениями. С их помощью, как и в большинстве других языков, осуществляются математические операции, проверки и изменения данных [7]. Однако, по сравнению с классическими языками, в разработанном языке имеется своя специфика данных операций, которая будет рассмотрена далее.

Основополагающим элементом языка, вокруг которого строятся его синтаксические и логические особенностями, являются поля, виды которых продемонстрированы в таблице 2. В структурах файлов они представляют собой непрерывные наборы байтовых значений различной длины. При этом отличительной особенностью их представления в языке является то, что одной синтаксической единицей осуществляется как чтение данных, так и их извлечение с одновременным преобразованием в требуемый формат согласно описанию.

Таблица 2. Форматы и описания различных видов полей

Тип команды поля	**Формат**	**Описание**
Прямое взятие	%имя поля% [%размер%:L/B]	Распарсить (произвести синтаксический анализ) поле или массив заданного размера в байтах с прямым / обратным порядком байтов и опционально указанием на наличие знака у числа. Возвращает ссылку на обработанный («распаршеный») объект.
Подусловное циклическое взятие	%имя поля% [%условие%]	Распарсить поле или массив с прямым / обратным порядком байтов и опционально указанием на наличие знака у числа, пока не будет встречено заданное условие. Возвращает ссылку на обработанный («распаршеный») объект.
Многомерное взятие	%имя поля%[][]...[]	Распарсить многомерный массив с заданными размерами или условиями. Возвращает ссылку на обработанный («распаршеный») объект

Таким образом, через операторы полей, обеспечивается извлечение данных сразу готовых к работе с ними, в том числе и по заданным условиям. Однако помимо выделения данных, требуется тем или иным способом использовать их значения. Для этого в языке есть механизм ссылок.

Ссылки используются для доступа к ранее «распаршеным» полям. По ссылке можно обращаться как к полю целиком, так и к его отдельным элементам. Более подробно различные виды ссылок описаны в таблице 3.

При этом необходимо отметить разное направление у ссылок и полей при обозначении подэлементов. Поля обрабатываются от внутреннего элемента к внешнему. В то же время ссылки от внешнего к внутреннему. Таким образом, порядок следования элементов представляет собой подобную последовательность:

%поле%[%количество байтов%][%размер внутреннего массива%]... [%размер внешнего массива%]

%ссылка%<%внешний индекс%>...<%внутренний индекс%> <%байт%><%бит%>

Таблица 3. Форматы и описания различных видов ссылок

Тип команды ссылки	**Формат**	**Описание**
Ссылка	%ссылка%	Простая ссылка, обращающаяся к одному объекту целиком.
Ссылка на элемент	%ссылка%<%индекс%>	Ссылка, обращающаяся к элементу объекта по индексу.
Ссылка на диапазон элементов	%ссылка%<%индекс%~%индекс%>	Ссылка на указанный диапазон элементов.
Ссылка на диапазон элементов с начала	%ссылка%<~%индекс%>	Ссылка на указанный диапазон элементов, начиная с начального индекса.
Ссылка на диапазон элементов до конца	%ссылка%<%индекс%~>	Ссылка на указанный диапазон элементов, заканчивая конечным индексом.
Ссылка на диапазоны элементов	%ссылка%<:%шаг%>	Ссылка на элементы, выделенные с определённым шагом, таким образом, будто бы промежуточные элементы отсутствуют в ссылаемом диапазоне.
Ссылка на диапазоны элементов в диапазоне	%ссылка%<%индекс%~%индекс%:%шаг%>	Ссылка на элементы, выделенные в определённом диапазоне с определённым шагом.
Ссылка на удовлетворяющие условия элементы	%ссылка%<%условие%>	Ссылка на элементы, удовлетворяющие заданному условию.
Ссылка на подэлементы	%ссылка%<><>...<>	Ссылка на подэлементы, больших уровней, выделяемые по тем же принципам, что и одноуровневые ссылки.
Объединение по ссылке	++	Последовательное объединение двух полей в одно.

Проверки существуют для обеспечения валидации и ветвления направления парсинга объекта. Путём сравнивания значений по ссылкам с требуемыми диапазонами, внутри программы можно определять дальнейшие действия.

Проверки полностью состоят из наборов операторов. Оператором считается как ссылка, так и действие над ней. Таким образом, имеется возможность рекурсивного использования значений в качестве элементов проверок, объединяя их в группы и наборы.

Однако поля, операторы и условия как таковые, сами по себе, как правило, не имеют смысла в контексте спецификаций форматов. В подавляющем числе случаев они сгруппированы в объекты. Каждый объект представляет собой определённым образом отформатированный блок данных, который может иметь параметры и различные размеры.

Для работы с такими элементами, концепция опционально параметрических объектов была реализована в синтаксисе языка, что показано в таблице 4.

Также в рамках языка была создана система расширений, поддерживающая как стандартный набор, так и пользовательский. Расширения стандартного набора включают в

себя элементы для обращения к особым элементам объектов, обозначения нулевого значения, доступа к индексам в циклическом парсинге полей, включения файлов, записи и чтения мета-информации, отправки сообщений и программе и многих других действий, не относящихся напрямую к процессу парсинга.

Таблица 4. Форматы и описания различных видов объектов

Тип объекта	**Формат**	**Описание**
Набор значений	{%имя поля...%}	Безымянный набор значений. Не ограничивает область видимости.
Массив наборов значений	{%имя поля...%}[]	Массив наборов значений. Не ограничивает область видимости. Каждый член независимо доступен по индексу набора.
Объект	$%имя объекта% := ...	Объект.
Перегружаемый объект	?$%имя объекта% := ...	Объект, для которого разрешена дальнейшая перегрузка.
Объект с собственным пространством имён	$%имя объекта%{...} := ...	Объект, обладающий локальным набором описаний объектов.
Параметризированный объект	$%имя объекта%(... , ... , ...) := ...	Объект с набором параметров, влияющих на его внутреннюю структуру.
Параметризированный объект с собственным пространством имён	$%имя объекта%(... , ... , ...){...} := ...	Объект, обладающий локальным набором описаний объектов с набором параметров, влияющих на его внутреннюю структуру.
Доступ к пространству имён объекта	%имя объекта%	Способ обратиться извне к подобъектам, содержащимся в локальном пространстве имён другого объекта.
Парсинг объекта	%имя экземпляра%$%имя объекта%	Инструкция для чтения объекта определённого типа и создания его экземпляра.
Парсинг массива объектов	%имя экземпляра%$%имя объекта%[]	Инструкция для чтения набора объектов определённого типа и создания их экземпляров.

В итоге, в процессе работы был получен программный комплекс для парисnга, анализа и валидации файлов различных форматов. Отличительными особенностями подхода являются гибкость, универсальность и многофункциональность. В его основе, по сравнению с классическим подходом, заключающемся в ручном написании алгоритма проверки файла на одном из классических языков, лежит упрощение процесса реализации валидации [8]. Это достигается благодаря непосредственно «заточенному» под вышеперечисленные цели синтаксису, компактному набору операторов и операций, заложенным в структуру языка и функционал транслятора оптимизациям и другим особенностям, как самого языка, так и системы его обработки.

Таким образом, для программиста, работающего на разработанной программно-алгоритмической системе, увеличивается качество и скорость построения механизма валидации файла, уменьшается вероятность ошибок в данном процессе, а также обеспечивается достаточная ясность и упорядоченность в конечном коде, позволяющая беспрепятственно изменять и расширять правила построенные в соответствии с определёнными стандартами или сторонними требованиями.

Список использованных источников

1. Carrier B. File System Forensic Analysis.- Pp. 2-3.- 2005
2. Shala L. Shala A. File Formats - Characterization and Validation // IFAC-PapersOnLine - Pp. 253-258, -2016, -Vol. 49, -No. 29.
3. Yara: The pattern matching swiss knife for malware researchers (and everyone else), Available online: https://virustotal.github.io/yara/ (accessed on 31 March 2020)
4. Kaitai Struct: declarative binary format parsing language, Available online: https://kaitai.io/ (accessed on 31 March 2020)
5. Microsoft Corporation, Word 2007: Rich Text Format (RTF) Specification, version 1.9.1, Available online: https://www.biblioscape.com/rtf15_spec.htm (accessed on 31 March 2020)
6. Microsoft Corporation, [MS-OLEDS]: Linking and Embedding (OLE) Data Structures, Available online: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-oleds/85583d21-c1cf-4afe-a35f-d6701c5fbb6f (accessed on 31 March 2020)
7. Kargara M. Isazadehb A. Izadkhahb H. Multi-programming language software systems modularization // Computers & Electrical Engineering -Vol. 80, (Dec. 2019).
8. Kokkea W. G.Siekb J. Wadler P. Programming language foundations in Agda // Science of Computer Programming, -Vol. 194, (Aug. 2020).