РЕЗУЛЬТАТЫ СРАВНИТЕЛЬНОГО АНАЛИЗАДЕКОМПИЛЯТОРОВ С ЦЕЛЬЮ ИССЛЕДОВАНИЯ БЕЗОПАСНОСТИ КОДА

Израилов К.Е.

Любое программное обеспечение может обладать недокументированными возможностями (НДВ). Наиболее критичными из них являются те, которые ставят под угрозу безопасность информационной системы. Поэтому поиск таких уязвимостей является актуальной задачей в системах с высоким требованием к уровню безопасности и отказоустойчивости. При отсутствии исходных кодов одном из способов поиска является исследование алгоритмов работы кода программного обеспечения. Для решения данной задачи применяется метод обратной разработки, или реверс-инжиниринга, в которым как правило, используется программа под названием декомпилятор. Она предназначена для трансляции исполняемого модуля в исходный код на языке программирования высокого уровня. Данное преобразование дает возможность провести изучение кода, его модификацию, восстановить алгоритм работы. Часто, под декомпилятором подразумевают получение исходного текста программы, который мог бы быть скомпилирован и использоваться в дальнейшем для его поддержки.

Был проведен анализ декомпиляторов, известных на данный момент, относительно возможности их применения для восстановления алгоритмов работы машинного кода с целью поиска НДВ. При этом задача получения декомпиляторами полностью компилируемого кода не ставилась. В сравнении учитывались процессоры исполняемого кода, текущая степень поддержки продукта, возможность его применения для выявления НДВ.

Результат сравнения декомпиляторов и их возможностей приведен в следующей таблице.

Таблица 1. Сравнение декомпиляторов

Назв	**Процес-**	**Под-**	**Примени-**	**Примечание**

ание	**сор**	**держ-ка**	**мость для поиска НДВ**	
Выходной язык – C				
Dede	x86	Нет	Нет	
REC	x86, Mips, PowerPC, mc68k	Да	Частично	Результат перегружен лишней для понимания работы алгоритмов информацией
Boomerang	x86, Sparc	Нет	Частично	Результат перегружен лишней для понимания работы алгоритмов информацией
DDC	x86	Нет	Частично	Результат перегружен лишней для понимания работы алгоритмов информацией
IDA	x86, ARM	Да	Да	Полноценный дизассемблер большинства процессоров. Декомпиляция реализована в виде плагина для x86 и ARM процессоров.
Выходной язык – VisualBasic и визуальные формы				
EMS Source Rescuer	x86	Нет	Нет	Не полноценный декомпилятор, восстановление визуальных форм, использует метаданные
VBRezQ	x86	Нет	Нет	Не полноценный декомпилятор, восстановление визуальных форм, использует метаданные
RaceEx6	x86	Нет	Нет	Не полноценный декомпилятор, восстановление визуальных форм,

				использует метаданные
ExDec	x86	Нет	Нет	Не полноценный декомпилятор, восстановление визуальных форм, использует метаданные
VBParser	x86	Нет	Нет	Не полноценный декомпилятор, восстановление визуальных форм, использует метаданные
P32Dasm	x86	Да	Нет	Не полноценный декомпилятор, восстановление визуальных форм, использует метаданные
VBDE	x86	Да	Нет	Не полноценный декомпилятор, восстановление визуальных форм, использует метаданные
Semi VB De-com-piler	x86	Да	Нет	Не полноценный декомпилятор, восстановление визуальных форм, использует метаданные
VB De-com-piler	x86	Да	Нет	Не полноценный декомпилятор, восстановление визуальных форм, использует метаданные
Выходной язык – FoxPro				
ReFox	вирт.маш.	Нет	Нет	Не полноценный декомпилятор, использует метаданные
Un-FoxAll	вирт.маш.	Нет	Нет	Не полноценный декомпилятор, использует метаданные
Выходной язык - .Net				

.Net Re-flector	вирт.маш.	Да	Нет	В коде присутствуют вся необходимые метаданные
Большое число других полноценных декомпиляторов по причине наличия всех необходимых метаданных. Исполняемый код выполняется на виртуальной машине. Не подходит для восстановления машинного кода.				
Выходной язык – Java				
JAD	вирт.маш.	Да	Нет	В коде присутствуют вся необходимые метаданные
Большое число других полноценных декомпиляторов по причине наличия всех необходимых метаданных. Исполняемый код выполняется на виртуальной машине. Не подходит для восстановления машинного кода.				

Согласно данному исследованию, большинство существующих декомпиляторов не предназначено для восстановления алгоритмов работы кода, работающего на реальном процессоре. Часть из них являются только дизассемблерами, часть давно не развиваются, в части ставится только цель получения компилируемого кода на выходе, а часть работают только с кодом, содержащим метаданные или выполняемых на виртуальных машинах (языки .VisualBasic, FoxPro, .Net, Java). Следовательно, их применения для поиска НДВ в коде не осуществимо.

Единственным подходящим для указанной цели продуктом является среда дизассемблирования IDA с плагином декомпиляции Hey-Ray. На данный момент IDA-плагин не является много-платформенным, т.к. поддерживает только x86 и ARM процессоры. Однако множество современных информационных систем используют иные типы процессоров, такие как PowerPC, MIPS. Примером этого может являтся коммуникационное оборудование Cisco. В этом случае IDA-плагин для декомпиляции не может быть применен.

Таким образом, задача по восстановлению алгоритмов работы кода с целью выявления НДВ для использования в современных системах не является полностью решенной. Исходя из ее актуальности, можно сделать вывод о необходимости разработки соответствующего метода и программного обеспечения.