УДК 004.492.2

АЛГОРИТМИЗАЦИЯ МАШИННОГО КОДА ТЕЛЕКОММУНИКАЦИОННЫХ УСТРОЙСТВ КАК СТРАТЕГИЧЕСКОЕ СРЕДСТВО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

ИЗРАИЛОВ К.Е.

АННОТАЦИЯ

В статье показана связь между национальной безопасностью государства и применяемыми в его инфраструктуре телекоммуникационными устройствами. Проанализирована динамика борьбы с уязвимостями в их программном обеспечении. Предложен способ увеличения эффективности поиска уязвимостей с помощью метода алгоритмизации машинного кода. Оценена применимость данного метода к различным типам уязвимостей.

Ключевые слова: информационная безопасность; уязвимость; метод алгоритмизации машинного кода; стратегическое средство.

ALGORITHMIZATION MACHINE CODE OF TELECOMMUNICATIONS DEVICES AS A STRATEGIC MEANS FOR INFORMATION SECURITY

IZRAILOV K.E.

ABSTRACT

The article shows the relationship between the national security of the state and its infrastructure used it telecommunication devices. The dynamics of the struggle with vulnerabilities in their software was analyzed. Is a way of increasing the efficiency of the vulnerability scan using the method of algorithmization machine code. Evaluated the applicability of the method to various types of vulnerabilities.

Keywords: information security; vulnerability; method of algorithmization machine code; strategic instrument.

Введение

Главнейшей функцией любого государства является обеспечение национальной безопасности, под которой понимается защищенность от внешних и внутренних угроз. В частности, возможной целью угроз является информация, несанкционированный доступ к которой может нанести ощутимый урон безопасности государства. Ярким примером служит Государственная тайна, понятие которой определено в российском законодательстве [1]. И если некую часть информацию теоретически и возможно защитить, полностью изолировав ее от враждебной среды, то остальная (большая) так или иначе подвержена хранению, передаче и обработке в открытой телекоммуникационной среде.

Поскольку целью государства является сохранение целостности общества (что является непрерывным по времени понятием), то и стратегия его развития должна планироваться не столько на ближайшее время, сколько на достаточно долгую перспективу. Следовательно, стратегические задачи и пути их достижения должны основываться не только на текущем состоянии зависимых областей, но и на экстраполяции динамики их развития.

Исходя из необходимости обеспечения информационной безопасности государства, напрямую зависящей от телекоммуникационной среды, можно утверждать о высоком приоритете задачи использования в ней безопасных телекоммуникационных

устройств (далее − ТКУ). Одной из определяющих характеристик таких устройств является наличие уязвимостей в их программном обеспечении (далее − ПО). Преследую же главную цель существования государства, решение задачи поиска уязвимостей должно учитывать прогнозы по развитию окружающей среды и не быть применимым лишь в конкретных ситуациях.

Таким образом, поддержание функции государства по обеспечению национальной безопасности напрямую зависит от эффективных средств обнаружения уязвимостей в ТКУ, применяемых в его инфраструктуре. Ситуация усложняется тем, что их ПО как правило является проприетарным и не поставляется с открытым исходным кодом, возможным для ручного анализа. При этом отсутствуют какие-либо гарантии того, что используемое в жизненно важных для государства структурах ТКУ априори не является специализированными для нанесения вреда его информационной безопасности, поскольку их производители давно стали транснациональными корпорациями и, следовательно, не обязаны следовать официальным государственным договоренностям. Таким примером может служить хорошо известная корпорация Cisco System Inc., являющаяся одним из крупнейших международных поставщиков высоких технологий и имеющая таких российских заказчиков (крайне важных для безопасности страны), как операторы сотовой связи, банки и госструктуры.

Производители ТКУ

Общее количество производителей ТКУ насчитывает несколько десятков компаний, однако доля рынка каждой из них сильно отличается. Структура рынка, описанная в отчете экспертом исследовательского центра Positive Research Дмитрием Курбатовым [2], приведена в следующее таблице.

Таблица 1. Структура рынка производителей ТКУ

Компания	Страна	Доля на рынке **(%)**
Cisco Systems	США	64
HP Networking	США	9
Alcatel-Lucent	Франция	3
Juniper Networking	США	2,3
Brocade	США	2,3
Huawei	КНР	1,8
Остальные		17,6

В случае России, приведенные компании будут несколько потеснены широко распространенной продукцией компании D-Link.

Для дальнейшего анализа безопасности ПО ТКУ выберем наиболее актуальных для России производителей, используя критерии популярности и влияния на информационную безопасность. Во-первых, таковыми будем считать обладающих наиболее высокой долей мирового рынка, а именно Cisco Systems и HP Networking. Во-вторых, являющихся хорошо известными в России, а именно D-Link. И в-третьих к списку производителей отнесем Китайскую компанию Huawei, сильно замешанную в скандалах, связанных с коммерческим шпионажем и разведывательной деятельностью. В частности, Конгресс США в 2008 году [3] ограничил последней доступ к сетям, передающим информацию государственного или военного характера из соображений национальной безопасности.

Уязвимости в ПО ТКУ

Для анализа уязвимостей ПО ТКУ будем использовать Национальную Базу Данных Уязвимостей или National Vulnerability Database (далее − NVD), являющуюся информационным хранилищем правительства США с автоматизированным управлением. Доступ к NDV может быть осуществлен через Интернет-сайт [4] с возможностью получения отчетов об уязвимостях согласно вводимым запросам. NVD имеет записи начиная с 1988 года и каждодневно пополняется. Отметим, что база данных содержит лишь обнародованные уязвимости, исключая скрытые производителями (например, из соображений сохранения имиджа компании). Впрочем, на общей динамике появления уязвимостей это не скажется, поскольку она определяется отношением между величинами, а не их абсолютными значениями.

Статистика найденных уязвимостей по выбранным производителям приведена в следующей таблице.

Таблица 2. Статистика найденных уязвимостей ТКУ согласно NVD

Год	Компания				Все компании
	Cisco Systems	**HP Networking**	**D-Link**	**Huawei**	
1988					0
1989					0
1990		1			1
1991		1			1
1992	2	0			2
1993	1	1			2
1994	0	8			8
1995	1	1			2
1996	0	19			19
1997	4	29			33
1998	6	11			17
1999	21	17			38
2000	16	36			52
2001	55	47	3		105
2002	65	58	5		128
2003	27	43	2		72
2004	36	29	3		68
2005	53	36	4		93
2006	69	39	10		118
2007	112	84	3	1	200
2008	93	72	5	0	170
2009	110	74	1	5	190
2010	154	117	3	0	274
2011	166	144	1	0	311
2012	159	85	3	3	250
2013	189	83	0	12	284

Для наглядности динамика количества уязвимостей производителей по годам представлена на следующем рисунке.

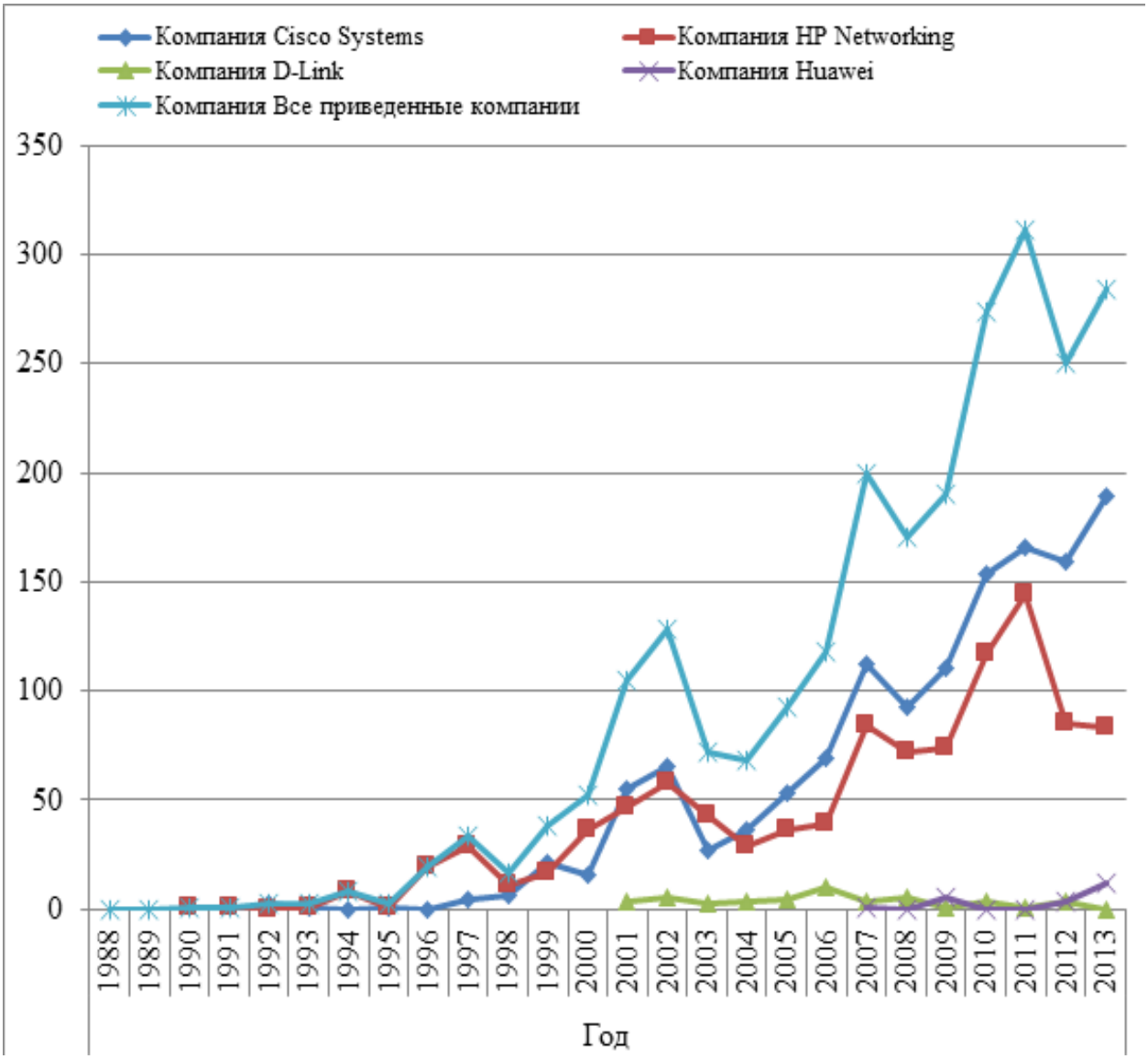

Рисунок 1 – Динамика найденных уязвимостей ТКУ согласно NVD

Отметим, что с учетом незавершенности текущего 2013 года, соответствующие ему уязвимости в приведенных данных хотя и не являются окончательными, однако могут быть использованы как нижняя граница итоговых значений.

Графический анализ приведенной динамики учтенных в NVD данных позволяет сделать следующие выводы. Во-первых, крупнейшие производители (Cisco Systems, HP Networking) так и не смогли добиться уменьшения возникающих уязвимостей или хотя бы остановить их рост. Во-вторых, остальные компании имеют предельно малое количество уязвимостей, что, скорее всего, связано с их «неохотой» в раскрытии реальной информации. И в-третьих, простейшая экстраполяция динамики позволяет утверждать о дальнейшем росте общего количества найденных уязвимостей.

Согласно анализу хорошо видно, что задача по более эффективному поиску уязвимостей остается крайне актуальной и по сей день. При этом прогнозируется еще большее ухудшение ситуации. Следовательно, дополнительно учитывая постоянный рост киберпреступности и значимость успешного ведения информационных войн, решение данной задачи носит перво-

очередной характер и должно являться частью стратегического развития любого государства, не ограничиваясь пределами России.

Поиск уязвимостей

Способы поиска уязвимостей можно разделить на две группы: основанные на ручном и автоматизированном подходах. Первый из них полностью полагается на опыт специалистов по информационной безопасности и характеризуется высокой степенью выявления сложных уязвимостей в коде при низкой эффективности относительно объема исследуемого кода. Второй же представляет собой использование специального ПО (как правило, сравнивающего части кода с заранее подготовленными шаблонами уязвимых мест) и обладает высокой скоростью обработки кода при возможности обнаружения лишь простых уязвимостей. Более редко применяется комбинация этих способов, как последовательная, так и параллельная.

При этом набирающий популярность способ «Fuzzing» [5] плохо применим для задачи поиска уязвимостей в ПО ТКУ, поскольку он затрагивает лишь область кода, ответственного за разбор и анализ входных данных. Бизнес-логика же ТКУ расположена в основном за фазой разбора входных данных и ее программная реализация с трудом может быть покрыта Fuzz-тестами.

В контексте затронутой проблемы, поиск уязвимостей различными способами осложняется следующими причинами. Во-первых, большинство продуктов телекоммуникаций не имеют открытых исходных кодов, что приводит к высокой трудоемкости применения ручного способа для ассемблера. Во-вторых, простые уязвимости обнаруживаются и исправляются самими компаниями во время фазы тестирования и применение одного лишь автоматизированного способа не даст значительных результатов. В-третьих, использование ТКУ в критичных для государства областях означает ненулевую вероятность сознательного внедрения в ПО уязвимостей враждебными структурами; поэтому данному риску должно быть уделено отдельное внимание при изучении кода специалистами.

Исходя из приведенных способов и сложностей их применения, решением задачи эффективного обнаружению уязвимостей может являться метод алгоритмизации машинного кода (далее − Метод), опубликованный в работах автора [6, 7].

Принцип работы Метода заключается в преобразование ассемблерного представления ПО к виду, понятному специалистами (например, к С-подобному языку) и адаптированному для его анализа с целью поиска уязвимостей (например, с отметками о подозрительных местах). А утилита, применение которой является основным этапом Метода, позволяет практически полностью автоматизировать данное преобразование. Можно считать, что утилита восстанавливает машинный код до такого вида, в котором ручной и автоматизированный способы поиска не только применимы, но и эффективны.

Метод (а точнее говоря, утилита автоматизации) может быть адаптирован к входному ассемблеру для большинства процессоров, применяемых в ТКУ (таких как PowerPC, MIPS, ARM). Также, выходное представление не обязано точно соответствовать существующим языкам программирования и содержать их «синтаксический сахар» [8]. Данные особенности сильно выделяют утилиту среди подобных, классически применяемых для реверс-инжиниринга, поскольку последние как настроены на какой-то конкретный процессоре выполнения, так и стремятся получить синтаксически верный и компилируемый код.

Применимость метода алгоритмизации кода

Хотя преимущества Метода очевидны, он является достаточно специализированным с точки зрения возможности обнаружения всех известных типов уязвимостей. Следовательно, для подтверждения его эффективности в части применимости необходимо определить лишь те уязвимости в ПО ТКУ, к которым он может быть применен. В качестве такого деления будем использовать каталог типов

уязвимостей в ПО из NVD, созданный на основе Общего Перечня Слабостей или Common Weakness Enumeration (далее − CWE) [9].

Каталог NVD состоит из следующих ID-разделов:

1) Authentication Issues − неправильная аутентификации пользователя;

2) Credentials Management − неправильное создание, хранение, передача или защита паролей и другой информации учетных записей;

3) Permissions, Privileges, and Access Control − неспособность соблюдения прав или других ограничений доступа к ресурсам, как и проблема управления привилегиями;

4) Buffer Errors − ошибка переполнения буфера, заключающаяся в занесении данных в область, недостаточную для их хранения;

5) Cross-Site Request Forgery (CSRF) − подделка межсайтовых запросов, заключающаяся в подмене сайта на враждебный с трансляцией запросов на защищенный;

6) Cross-Site Scripting (XSS) − межсайтовый скриптинг, заключающийся во внедрении выполняемых на стороне клиента вредоносных скриптов в выдаваемую системой страницу;

7) Cryptographic Issues − проблема, связанная с отсутствием шифрования и неверной реализацией его алгоритмов (слабые ключи или сертификаты управления, зависимая генерации случайных чисел и т.п.);

8) Path Traversal − отсутствие защиты от доступа к запрещенным файлам системы при помощи использования во вводимых путях команды перехода в каталог выше (например поднятие на уровень выше «..» в большинстве операционных систем);

9) Code Injection − выполнение кода злоумышленника в файлах, загружаемых в систему;

10) Format String Vulnerability − использование злоумышленником в качестве ввода данных в систему параметров форматирования строки, приводящих к неверному разбору последней определенными функциями (например, использование символов «%s» в качестве вводимых данных в случае их разбора функциями ввода/вывода с поддержкой формата строки sprint/sscanf);

11) Configuration − ошибка конфигураций, не связанная с паролями, разрешениями и другими функциями системы;

12) Information Leak/Disclosure − утечка или раскрытие информации, являющейся конфиденциальной (пароли, биометрическая информация и т.п.);

13) Input Validation − отсутствие проверки вводимых данных на их корректность и соответствие спецификации приложения (может перекрываться с другими типами уязвимостей);

14) Numeric Errors − ошибка работы с числами, такая как их переполнение, неверная знаковость, сокращение и т.п.;

15) OS Command Injections − возможность запуска небезопасных функций системы, таких как «system()», посредством контролируемого пользователем ввода;

16) Race Conditions − ошибка проектирования многопоточной системы, при которой ее работа зависит от того, в каком порядке выполняются части кода (например, состояние ресурса может изменится с момента последней проверки до момента доступа к нему);

17) ResourceManagementErrors−ошибка нехватки ресурсов системы, возникающая при утечках памяти, чрезмерной нагрузке на систему злоумышленником или в других подобных случаях;

18) SQL Injection − отсутствие защиты от внедрения враждебного кода в SQL-запросы;

19) Link Following − отсутствие защиты от доступа к запрещенным файлам системы путем использования символических или «жестких» ссылок;

20) Design Error − ошибка в проектировании системы;

21) Other − другие уязвимости;

22) Not in CWE – уязвимость, не рассматриваемая в CWE;

23) Insufficient Information − недостаточное количество информации об уязвимости.

Последние три типа уязвимости (21−23) приниматься во внимание не будут

по причине отсутствия их характеристик и определений.

Согласно принципу работы Метода, его применимость к типизированным уязвимостям из каталога NVD с обоснованиями приведена в следующей таблице.

Таблица 3. Применимость Метода к типам уязвимостей из каталога NVD

ID	Применимость	Обоснование/Результаты работы
1	Частичная	Восстановленные алгоритмы будут описывать логику аутентификации
2	Полная	Возможные уязвимости при работе с учетными записями хорошо видны по восстановленному коду, оперирующему такими понятиями как объект (например, пароль) и функция (например, алгоритм сериализации)
3	Частичная	Контроль управления привилегиями и ограничениями в большей степени относится к особенностям проектирования системы и поэтому не может быть проанализирован с помощью Метода в полном объеме
4	Полная	Возможность выхода данных за диапазоны выделенного буфера как правило возникает в языках не достаточно высокого уровня (таких как C), на которых и базируется результирующее представление Метода
5	Отсутствует	Метод не подходит для поиска уязвимостей в Web-приложениях
6	Отсутствует	Метод не подходит для поиска уязвимостей в Web-приложениях
7	Полная	Восстановленные алгоритмы будут описывать логику работы криптографии
8	Частичная	Метод мало подходит для поиска уязвимостей в файловой системе, однако может быть применен для анализа механизма проверки корректности входных данных
9	Полная	Восстановленные алгоритмы будут описывать логику работы с файлами, загружаемыми в систему
10	Полная	Восстановленные алгоритмы будут описывать работу с функциями форматированного ввода/вывода строки
11	Отсутствует	Метод не включает проверку корректности конфигурацию системы
12	Полная	Восстановленные алгоритмы будут описывать логику работы с конфиденциальной информацией
13	Частичная	Хотя восстановленные алгоритмы и будут описывать логику работы механизма проверки корректности вводимых данных, однако Метод не включает соответствие этих данных спецификации приложения
14	Частичная	Хотя восстановленные алгоритмы и будут описывать логику работы с числовой информацией, однако описание не будет достаточно детальным
15	Частичная	Метод мало подходит для поиска уязвимостей в файловой системе, однако может быть применен для анализа механизма проверки корректности входных данных
16	Полная	Вид восстановленных алгоритмов, как набора функций, глобальных переменных и взаимосвязи между ними достаточно хорошо подходит для анализа работы многопоточной системы
17	Частичная	Хотя восстановленные алгоритмы и будут описывать логику работы с ресурсами системы, однако сам по себе Метод не имеет этапов тестирования нагрузоустойчивости последней
18	Отсутствует	Метод мало подходит для поиска уязвимостей в SQL-запросах
19	Частичная	Метод мало подходит для поиска уязвимостей в файловой системе, однако может быть применен для анализа механизма проверки корректности входных данных
20	Полная	Вид восстановленных алгоритмов, как набора функций, глобальных переменных и взаимосвязи между ними достаточно хорошо подходит для воссоздания архитектуры отдельных модулей системы

Для определения доли каждой уяз-вимостей в общем количестве приведем статистику NVD по каждому из ее типов (используя ID) для лидера рынка произво-дителей ТКУ – Cisco Systems.

Таблица 4. Статистика уязвимостей по типам для Cisco Systems согласно NVD

Год	Тип уязвимости (ID)																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1992																				
1993																				
1994																				
1995																				
1996																				
1997												1								
1998												0								
1999												1								
2000												0	1							
2001	1											0	1				1			
2002	0			3			4			1		0	4				1			
2003	1			0			3			0		1	1				0			0
2004	1			0			11			0		0	2				1			0
2005	1			0			8			0	1	0	0				1			1
2006	0		1	2			5			0	1	0	1				3			0
2007	3	2	6	6		4	4			0	1	3	7	1			2	1		2
2008	6	3	7	8	1	3	27			0	4	4	21	0			13	1		3
2009	3	5	6	8	5	11	4	5	1	0	2	5	7	0			6	0		4
2010	1	9	14	6	1	8	3	3	1	0	1	5	18	0	1		6	3	1	1
2011	7	6	16	11	1	5	11	4	5	0	3	5	5	0	7	1	57	4	0	1
2012	5	0	10	29	3	4	8	1	6	0	2	6	29	1	3	3	35	2	0	3
Всего	29	25	60	73	11	35	88	13	13	1	15	31	97	2	11	4	126	11	1	15

Исходя из общего количество типов уязвимостей, ставя в соответствие применимости Метода к каждому из них свой коэффициент (100% − для полной, 50% − для частичной и 0% − для отсутствующей), а также учитывая количество найденных уязвимостей каждого типа, можно рассчитать средневзвешенное арифметическое значение применимости Метода с помощью следующего выражения:

$$\bar{A} = \frac{\sum_{i=1}^{N} A_i \times V_i}{\sum_{i=1}^{N} V_i},$$

где N – количество всех типов уязвимостей, V_i – количество уязвимостей типа i, A_i – коэффициент применимости для типа i.

Вычисление выражения по приведенным ранее данным (см. табл. 3−4) дает значение $\bar{A}$=63.5%, соответствующее числовому эквиваленту применимости Метода (в случае уязвимостей в ПО Cisco System). При этом, Метод абсолютно неприменим лишь для таких специфичных типов уязвимостей, как Web-приложения и SQL-запросы, которые в основном зависят от разработчиков сайтов и баз данных, а не от ПО ТКУ. Для всех остальных типов Метод может быть применен полностью или частично.

Таким образом, применимость Метода

для поиска уязвимостей в ПО ТКУ является бесспорной.

Вывод

Одним из фундаментов жизнедеятельности государства является функционирующее в нем информационное пространство. Реалии же окружающего мира дают четко понять, что используемая госу-дарством информация находится под постоянной угрозой нарушения ее безопасности. При этом, как современное состояние способов защиты информации, так и динамика их развития не позволяют говорить о каких либо видимых улучшениях. Следовательно, одним из важнейших путей развития государства должен являться рост эффективности борьбы с уязвимостями в ПО ТКУ, ответственном за хранение, передачу и обработку информации. Возможным решением может быть предложенный метод алгоритмизации машинного кода (его применимость наглядно показана на примере ПО компании Cisco System).

С учетом всего вышеизложенного, метод алгоритмизации машинного кода телекоммуникационных устройств можно считать стратегическим средством обеспечения информационной безопасности государства.

Список литературы

1. «О государственной тайне»: Закон РФ от 21 июня 1993 г. № 5485-I.

2. Популярное сетевое оборудование и статистика уязвимостей. [Электронный ресурс]. – Режим доступа: http://www.lesc.ru/index.php/analitika/474-populjarnoe-setevoe-oborudovanie-i-statistika.

3. ANNUAL REPORT TO CONGRESS: Military Power of the People's Republic of China 2008. [Электронный ресурс]. – Режим доступа: http://www.defense.gov/pubs/pdfs/China_Military_Report _08.pdf.

4. National Vulnerability Database. [Электронный ресурс]. – Режим доступа: http://nvd.nist.gov/.

5. Fuzzing. [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/Fuzzing.

6. Буйневич М.В., Израилов К.Е. Метод алгоритмизации машинного кода телекоммуникационных устройств.// Телекоммуникации, № 12.– 2012.– С. 2–6.

7. Буйневич М.В., Израилов К.Е. Автоматизированное средство алгоритмизации машинного кода телекоммуникационных устройств.// Телекоммуникации, № 6.– 2013.– С. 2–9.

8. Синтаксический сахар. [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/Синтаксический_ сахар.

9. Common Weakness Enumeration. [Электронный ресурс]. – Режим доступа: http://cwe.mitre.org/.