ОБОЗРЕНИЕ
ПРИКЛАДНОЙ И ПРОМЫШЛЕННОЙ
Том 25 МАТЕМАТИКИ Выпуск 3
2018

С. С. Мигалин, **В. О. Миронкин** (Москва, Национальный исследовательский университет, Высшая школа экономики)**. Об энтропии исходных кодов программ в C++.**

Введение.

Одной из потенциальных слабостей программно реализованных приложений является возможность внедрения вредоносных закладок в исполняемый программный код. Подобная модификация текста программы может представлять собой как пассивную угрозу (передача скрытой информации по контролируемому злоумышленником каналу связи), так и активную угрозу (реализация вредоносной части кода в интересах злоумышленника — например, компьютерные вирусы).

Таким образом, наиболее актуальными в рассматриваемой проблематике становятся задачи, связанные как с выявлением самого факта использования вредоносного кода, так и с непосредственным обнаружением участка программы, использующего данный код. Для решения подобного класса задач необходим целый спектр подходов и инструментов, в частности, таких, как непосредственное дезассемблирование (анализ исполняемого кода), вероятностно-статистический анализ, лексический анализ и т. д. При этом очевидно, что для реализации методов выявления вредоносных участков программ необходимо наличие теоретико-вероятностной модели формирования исполняемого кода.

Настоящая работа посвящена изучению некоторых информационных характеристик языка C++ (одного из самых популярных языков программирования), на основе которых могут быть построены критерии выявления факта внедрения опасной информации в исполняемый код программ.

Для проведения экспериментальных исследований в качестве обрабатываемого материала были использованы тексты корректно работающих программ с открытым исходным кодом, представленным на сервисе GitHub. При этом были выбраны самые популярные репозитории, содержащие коды в одном из следующих стандартов: C++98, C++11, C++14, а именно,

- **C++98**: 1d-cellular-automata, brick-break, btree, cpp_98, Cpp-fast-delegates, cpp-mmf, cpp_stl_98, cpp-tuple, ct_crypt, dectable, docopt.cpp98, EmbeddedPlatformLite, epitech-rush-cpp-hell, etl-98, filter-iterator, functional, Glumer, ivl, myrrh, nJson, Number, PowerWhois, Rcpp98, sauce, sentiment-classifier, thread_pool, tokenizer, travis_qmake_gcc_cpp98_qt5, unit-test, x2boost.
- **C++11**: asyncplusplus, ATen, beast, botan, cereal, cosmos, docopt.cpp, json11, LeetCode, libgo, nanomsgxx, opendht, origin, pybind11, served, socket.io-client-cpp, sol, ThreadPool, thread-pool-cpp, tungsten, Turbo, uvpp, wgtcc, word2vec.
- **C++14**: aura-bot, cat, constexpr-8cc, continuable, C-PCB, cpp14-appendix, cpp-frp,cpp-sort, di, EntityPlus, fillwave, fp, gamgee, gf, jeayeson, libsourcey, measurement-kit, metaverse, mettle, qhttp, QuickQanava, Reflex, SimpleRenderEngine, static_math, tiny-dnn, tmxlite, transducer, trompeloeil, vpp, xlnt.

З а м е ч а н и е 1. Стандарт C++98 был выбран в качестве исходного стандарта C++, а C++11 и C++14 — как надстройки, использующие новый синтаксис.

Предварительная обработка исходных текстов программ осуществлялась в два этапа:

1. Обработка соответствующего стандарта препроцессором компилятора, позволяющая удалить комментарии авторов, исключить некорректные файлы, а также проверить исходные тексты на ошибки.
2. Удаление символов национальных алфавитов, пробелов, отступов и переносов строк, не влияющих на корректность работы программ.

В итоге объем анализируемых данных, сформированных после данной обработки составил 50 536 678 символов.

1. Теоретико-вероятностная модель языка C++.

C++ — компилируемый, статически типизированный язык программирования общего назначения, широко используемый для разработки программного обеспечения [2].

При написании исходных текстов стандарт языка C++ предусматривает использование 96 различных символов, среди которых буквы латинского алфавита, имеющие различное семантическое значение в зависимости от регистра, арабские цифры, а также специальные символы, знаки пробела и переноса строки.

Введем ряд используемых далее обозначений и определений:

- A_{C++} — алфавит, используемый в языке C++, $|A_{C++}| = 95$ (при проведении экспериментов знак переноса строки не учитывался, поскольку текст корректно работающей программы может быть написан в одну строчку).
- C_l — множество слов длины $l \in \mathbb{N}$, записанных в алфавите языка C++, $|C_l| = 95^l$.
- $\overline{p}_{C++}$ — вероятностное распределение, заданное на символах A_{C++}.

Согласно теории [1] мы вправе говорить о дискретном источнике сообщений $\left(A_{C++}, \overline{p}_{C++}\right)$.

О п р е д е л е н и е 1. Энтропией дискретного источника сообщений на один знак называется величина

$$H_l = -\tfrac{1}{l} \sum_{\left(a_{i_1},\ldots,a_{i_l}\right)\in C_l} \mathbf{P}\left(a_{i_1},\ldots,a_{i_l}\right) \log \mathbf{P}\left(a_{i_1},\ldots,a_{i_l}\right). \tag{1}$$

О п р е д е л е н и е 2. Шаговой энтропией дискретного источника сообщений называется величина

$$H^{(l)} = - \sum_{\left(a_{i_1},\ldots,a_{i_l}\right)\in C_l} \mathbf{P}\left(a_{i_1},\ldots,a_{i_l}\right) \log \mathbf{P}\left(a_{i_l}/a_{i_1},\ldots,a_{i_{l-1}}\right). \tag{2}$$

З а м е ч а н и е 2. Объем анализируемого текста не позволяет считать исследуемый источник сообщений, генерирующий символы языка C++, эргодическим. По этой причине в работе экспериментально определены значения параметра l, используемого в (1) и (2), для которых соответствующие значения энтропии являются адекватными.

Далее приведены результаты экспериментальных исследований ряда информационных характеристик языка C++.

2. Основные результаты.

В табл. 1 представлены приближенные (с точностью 10^{-2}) значения частот встречаемости символов языка C++, полученные на объеме 50 536 678 символов.

Из табл. 1 видны некоторые особенности синтаксиса языка: частое использование парных скобок, символа нижнего подчеркивания для обозначений имен переменных, нуля как префикса шестнадцатеричных чисел, точки с запятой как символа конца выражения и двоеточия, которое используется для обозначения квалифицированных имен.

Таблица 1. Частоты встречаемости символов в языке C++

символ	**частота, %**	**символ**	**частота, %**	**символ**	**частота, %**	**символ**	**частота, %**
” ”	11,50	f	1,14	C	0,45	G	0,19
e	6,51	2	1,07	{	0,45	H	0,18
t	5,30	=	1,03	}	0,45	V	0,13
a	3,78	T	1,00	&	0,42	X	0,10
r	3,53	7	0,98	R	0,39	q	0,08
s	3,50	b	0,98	I	0,37	W	0,08
n	3,47	y	0,87	[	0,36	K	0,07
i	3,19	g	0,85	]	0,36	j	0,07
,	3,14	6	0,76	P	0,35	\	0,07
o	2,83	3	0,74	+	0,34	!	0,06
_	2,36	5	0,73	k	0,34	Y	0,05
c	2,32	.	0,72	L	0,32	Q	0,05
p	2,26	h	0,72	N	0,31	‘	0,04
d	2,11	4	0,70	–	0,30	|	0,04
0	2,05	v	0,68	*	0,29	Z	0,03
l	2,02	<	0,67	/	0,28	J	0,02
m	1,84	U	0,64	O	0,27	%	0,01
1	1,76	>	0,62	w	0,27	?	0,01
)	1,69	E	0,62	D	0,25	~	0,01
(	1,69	”	0,61	M	0,24	^	0,01
u	1,65	8	0,59	#	0,23	$	0,00
;	1,65	S	0,52	B	0,22		0,00
:	1,48	9	0,47	F	0,22	‘	0,87
x	1,23	A	0,46	z	0,22		

Наряду с частотами отдельных символов были экспериментально получены и частоты встречаемости l-грамм для $l \in \{1, \ldots, 20\}$. Так наиболее вероятными и «осмысленными» являются следующие последовательности из C_l:

- $l \in \{2, 3\}$ — префикс шестнадцатеричного числа ‘0x’ и предлог ‘in’.
- $l = 4$ — последовательности ‘type’, ‘name’, которые по отдельности не образуют никаких ключевых слов C++, но, написанные слитно, образуют конструкцию ‘typename’, являющуюся частью описания шаблонов C++. Данная конструкция используется при каждом определении любой шаблонной функции [4].
- $l = 5$: ‘const’ — квалификатор константного значения, ‘std::’ — обращение к пространству имен стандартной библиотеки C++.
- $l = \{6, 7\}$ — ‘const ‘, ‘return’ — оператор возврата управления.
- $l = 8$ — ‘typename’, ‘ return ‘.
- $l = 9$ — ‘typename ’, ‘ 16777215’ — константа, которая используется для обозначения количества цветов, используется в библиотеках для работы с изображениями ($256^3 - 1$).
- $10 \leqslant l \leqslant 20$ — последовательность ‘traits::input_parameter’, которая является описанием типа данных, используемого в производительных библиотеках C++, связанных со статистическими вычислениями [4].

На основе выражений (1), (2), где в качестве оценки вероятности использовалась частота встречаемости l-грамм, были экспериментально получены значения соответствующих информационных характеристик. Так в табл. 2 представлены приближенные (с точностью 10^{-2}) значения энтропии на знак H_l и шаговой энтропии $H^{(l)}$ языка C++, выраженные в битах, где $l \in \{1, 2, \ldots, 10\}$.

Таблица 2. Значения характеристик H_l и $H^{(l)}$

l	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
H_l	5,52	4,84	4,23	3,69	3,25	2,90	2,62	2,38	2,19	2,02
$H^{(l)}$	5,52	4,17	3,00	2,06	1,50	1,14	0,92	0,75	0,62	0,51

Для оценки значения параметра $l \in \mathbb{N}$, позволяющего адекватно на заданном объеме материала оценить значения шаговой энтропии и энтропии на знак, был сгенерирован текст объема 50 536 678 символов по следующему правилу:

1. Исходный текст был разделен на 20 равных по объему частей, которые впоследствии были пронумерованы:
$$\{a_1^{(i)}, a_2^{(i)}, a_3^{(i)}, \dots\}, \qquad i \in \overline{1,20};$$
2. Последовательно из каждой части по порядку (с периодом 20) извлекалось по одному символу, и формировался тест M объема 50 536 678 :
$$M \equiv \{a_1^{(1)}, \dots, a_1^{(20)}, a_2^{(1)}, \dots, a_2^{(20)}, a_3^{(1)}, \dots, a_3^{(10)}, \dots\}.$$

О п р е д е л е н и е 3. Источник сообщений, вырабатывающий в единицу времени символы алфавита независимо от ранее выработанных называется дискретным источником без памяти (далее — ДИБП).

З а м е ч а н и е 3. В рамках проведенных исследований сформированный текст M рассматривался как реализация ДИБП. При этом корреляция символов, следующих друг за другом на расстоянии 20 знаков не учитывалась, поскольку длины последовательностей, используемых в выражениях (1), (2), для заданного объема заведомо меньше 20).

Согласно [3] для ДИБП значение энтропии на знак и шаговой энтропии являются величинами постоянными и равными $H_1 = -\sum\limits_{a_i \in A_{C++}} \mathbf{P}(a_i) \log \mathbf{P}(a_i)$. По этой причине для текста M было экспериментально определено значение величины

$$l_{\max} = \max_{l \in \mathbb{N}} \{l | H_1 = \dots = H_l; H^{(1)} = \dots = H^{(l)}\} = 4, \tag{3}$$

при котором выполняется указанное свойство (см. рис. 1).

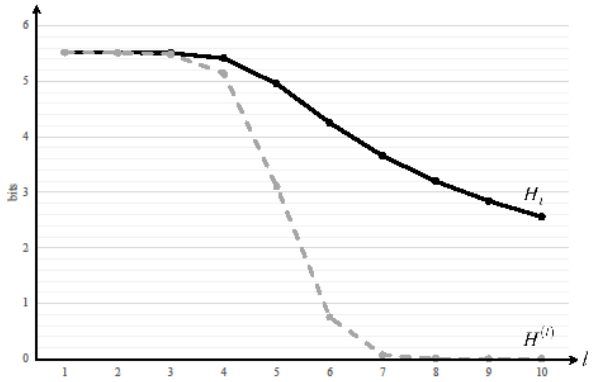

Рис. 1. Графики поведения характеристик H_l и $H^{(l)}$ для текста M

Соответственно, для длин $l \in \{1, \dots, 4\}$ были получены приближенные значения шаговой энтропии и энтропии на знак для трех надстроек стандарта C++ (см. табл. 3).

Таблица 3. Значения характеристик H_l и $H^{(l)}$ языков C++98, C++11, C++14

Размер l-граммы	**C++98**		**C++11**		**C++14**	
	H_l, биты	$H^{(l)}$, биты	H_l, биты	$H^{(l)}$, биты	H_l, биты	$H^{(l)}$, биты
1	5,38	5,38	5,48	5,48	5,56	5,56
2	4,50	3,62	4,72	3,97	4,90	4,25
3	3,78	2,32	4,06	2,73	4,30	3,08
4	3,21	1,53	3,49	1,79	3,75	2,11

На рис. 2 изображена зависимость энтропии на знак H_l от величины l для соответствующих надстроек языка C++.

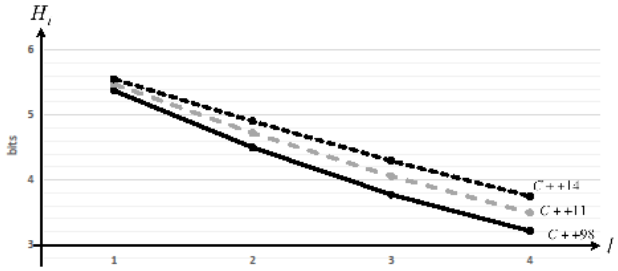

Рис. 2. Графики поведения H_l для трех стандартов C++

Увеличение энтропии при переходе к более новым стандартам языка C++ (и это видно из графика) объясняется введением существенно новых синтаксических конструкций в более поздних надстройках языка.

СПИСОК ЛИТЕРАТУРЫ

1. *Духин А. А.* Теория информации. М.: Гелиос АРВ, 2007.
2. *Лафоре Р.* Объектно-ориентированное программирование в C++. СПб.: Питер, 2012.
3. *Чечета С. И.* Введение в дискретную теорию информации и кодирования. М.: МЦНМО, 2011.
4. *Eddelbuettel D.* Seamless R and C++ Integration with Rcpp, Springer, 2013, p. 68.