УДК 004.52
А.О.Нічепорук, О.С.Савенко
Хмельницький національний університет

МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ ПОЛІМОРФНИХ ВІРУСІВ

В роботі розглянуто етапи життєвого циклу поліморфного вірусу. Наведено класифікацію поліморфних вірусів. За допомогою теорії формальних граматик розроблено моделі життєвого циклу кожного рівня поліморфізму згідно з класифікацією.

Ключові слова: *поліморфний вірус, розшифровщик, формальна граматика, алфавіт.*
Форм.15. Рис.6. Літ 9.

Вступ

Поліморфні віруси на сьогодні залишаються одними з найбільш небезпечних вірусів, що представленні в інформаційному середовищі. Поліморфізм це техніка, що спрямована на запобігання виявлення шкідливого коду антивірусними засобами.

Метою даної статті є аналіз техніки поліморфізму, на основі якої побудовані поліморфні віруси, а також проведення класифікації поліморфних вірусів. На основі класифікації запропоновано моделі життєвого циклу кожного рівня поліморфізму за допомогою формальних граматик.

Актуальністю роботи є опис формальних моделей, що лягли б в основу методу виявлення поліморфного шкідливого програмного коду.

Поліморфні віруси, їх життєвий цикл

Поліморфний вірус – вірус, що відноситься до класу шифрованих вірусів і який складається з основного тіла вірусу, розшифровщика та поліморфного генератора[1,2,3,4].

Поліморфний генератор – підпрограма поліморфного вірусу, що виконує функції видозмінення вірусу при розповсюдженні, що складається з блоку шифрування корисного навантаження та безпосередньо генератора відповідних дескрипторів.

Життєвий цикл поліморфного вірусу зображений на рис.1. При старті зараженої програми вірусний поліморфний дескриптор розшифровує основне тіло вірусу та передає йому керування[4].

Далі основний вірусний код виділяє ділянку пам'яті у верхніх адресах, копіює в нього власний код і передає йому керування. Потім він відновлює код зараженого файлу в програмному сегменті (для EXE-файлів також проводить налаштування адрес елементів, що переміщаються) і приступає до безпосереднього впровадження у пам'ять своєї резидентної копії.

Після свого розміщення у пам'яті вірус виконує ряд підготовчих операцій для виділення пам'яті під своє тіло і проводить перемикання процесора в захищений режим роботи з найвищим рівнем пріоритету – режим супервізора. В процесі своєї роботи вірус може перехоплювати ряд системних ресурсів, зокрема, API-функції, переривання, функції, що експортовані ядром ОС. Перехоплення API-функцій виконується через таблицю імпорту функцій. Дана таблиця заповнюється при завантаженні DLL в процес і в ній прописуються адреси всіх імпортованих функцій, які можуть бути необхідні процесу. Відповідно, вірус знаходить таблицю імпорту, в ній - функцію, яку він хоче перехопити, зберігає там адресу (покажчик на тіло функції), після чого розміщує туди покажчик на свою функцію. Після того, забороняє вивантаження DLL на час перехоплення (наприклад, DllCanUnloadNow повинна повертати false), щоб у процесі роботи DLL не була вивантажена, а адреса перехоплення не стала хибна.

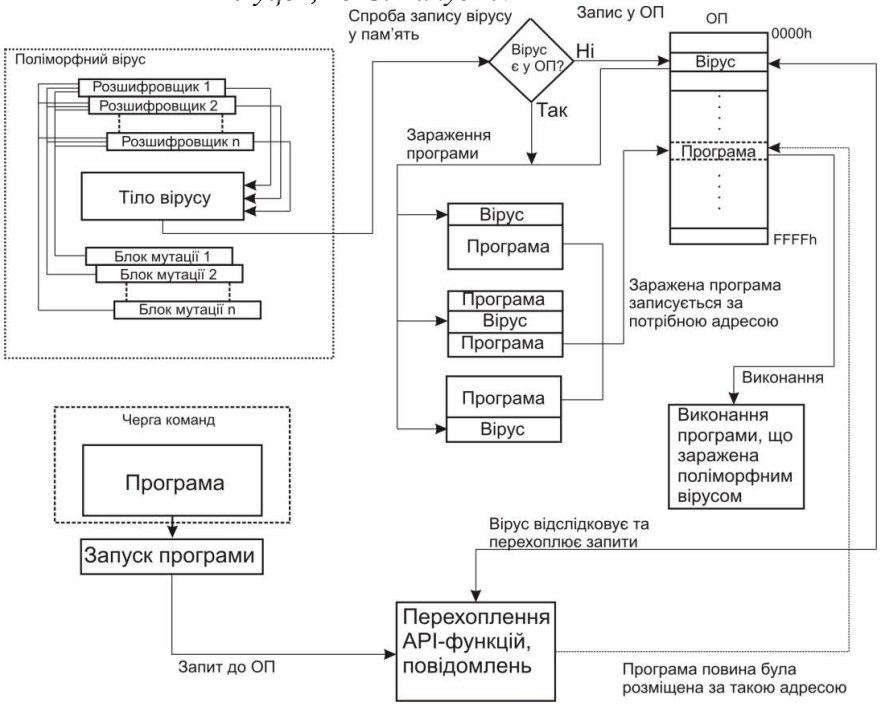

Рис. 1 . Загальний цикл життєвого циклу поліморфних вірусів
Авторська розробка

Нехай є мова, що описується формальною граматикою, яка складається з наступної четвірки об'єктів: G=(T, N, P, S), де[8]:

T – алфавіт термінальних символів (терміналів). Літери цього алфавіту називаються термінальними символами, з них будуються ланцюжки, що породжуються граматикою.

N – алфавіт не термінальних символів (не терміналів), що не перетинається з T. Літери цього алфавіту використовуються при побудові ланцюжків.

P – множина правил висновку чи породжуючих правил наступного виду: $\alpha \to \beta$ де α і β - ланцюжки побудовані з літер алфавіту $N \cup T$, який має назву повний алфавіт (словник) граматики G. Правилами будемо вважати: $P \subseteq (T \cup N)^* \times (T \cup N)^*$, такі що $(u,v) \in R \Rightarrow u \notin T^*$

S – початковий символ граматики, $S \in N$.

Мовою, що породжена граматикою G=(T, N, P, S), називається множина $L(G)=\{\alpha \in T^* \mid S \Rightarrow \alpha\}$.

Іншими словами, L(G)- це всі ланцюжки в алфавіті T, що виведені з S за правилами P.

Вперше зв'язок між поліморфізмом та формальними граматиками показаний Козахом[6]. Автором було доведено, що поліморфний вірус може бути описаний за допомогою формальної граматики.

Алфавітом формальної граматики будемо вважати множину інструкцій x86, яка є досить великою, проте скінченою. Позначимо алфавіт скінчених команд x86 через UL. Відомо також, що команди x86 мають внутрішні правила, адресування та ін, тому будемо вважати команди, наприклад mov [eax],esi та mov [eax+125],ebx еквівалентними[6].

Розрізняють шість рівнів поліморфізму[5]. Розглянемо детально кожен рівень та представимо моделі життєвого циклу формальною граматикою.

Поліморфні віруси першого рівня

Поліморфні віруси можуть бути класифіковані на основі складності коду підпрограми коду розшифровки вірусу [5,9]. Вперше така класифікація була проведена Аланом Соломоном і продовжена Васеліном Бончевом.

До цього рівня належать поліморфні віруси, що проводять розшифровку основного тіла вірусу, вибираючи один з декількох постійних розшифровщиків, в яких міститься постійний код.

Для реалізації таких поліморфних вірусів створюється деяка кількість розшифровщиків, вибирається випадковий розшифровщик і зашифровує тіло вірусу разом з рештою розшифровщиками. В кожному наступному поколінні розшифровщик розшифровує вірус, передає управління основному тілу, а перед втіленням вибирає випадковий розшифровщик, яким і

зашифровує його. Код вірусу повністю змінюється і "візуально" розпізнати заражений файл вже не є можливим.

Оскільки кількість розшифровщиків завжди є скінченним числом (навіть якщо воно дуже велике), сигнатурний пошук залишається досить ефективним заходом протидії. Антивірус заносить в базу сигнатури всіх розшифровувачів. Проте, розшифровувачі різних вірусів зазвичай дуже схожі і тому в їх детектуванні немає ніякої користі. Антивірус змушений залучати емулятор, що імітує виконання розшифровувача і розшифровує основний вірусний код. Якщо розшифровщик містить антивідлагоджувальні команди або використовує машинні інструкції, що не підтримуються емулятором, то антивірус не зможе виявити вірус. Чим більше антивідлагоджувальних прийомів містить розшифровувач, тим вище його унікальність і, отже, надійніше детектування.

Приклад розшифровувача на основі XOR зображено наступною послідовністю інструкцій:

```
MOV     ESI, offset body_begin
    MOV     EDI, ESI
    MOV     ECX, offset body_end - body_begin
my_begin:
    LODSB
    XOR     AL, 66h
    STOSB
LOOP my_begin
body_begin:
; // тіло вірусу зі всіма іншими шифровщиками
body_end:
```

Опишемо даний рівень формальною граматикою.

Нехай є формальна граматика G1=(T, N, P, S), яка складається:

T={a,b,x,y} – термінальний алфавіт, що містить інструкцій x86, a,b – команди програми, x,y – команди розшифровщика;

N={Q1,Q2} – нетермінальний алфавіт де Q1, Q2 – рекурсивні функції вибору розшифровщика, α – кінець ланцюжка;

P – множина правил вибору розшифровщика;

S – початковий символ.

Запишемо правила для граматики G1:

$$P = \begin{cases} S \to aS \mid bS \mid xQ1 \mid yQ2 \\ Q1 \to xQ1 \mid xS \mid x\alpha \\ Q2 \to yQ2 \mid yS \mid y\alpha \\ \alpha \to "" \end{cases}$$

В результаті може бути отриманий такий ланцюжок виведення baxxxauy. Життєвий цикл поліморфного вірусу першого рівня представлено на рисунку 2.

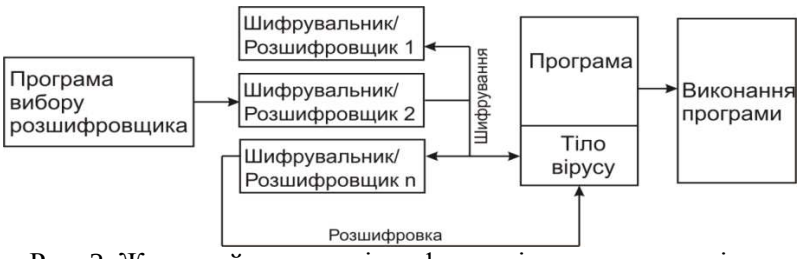

Рис. 2. Життєвий цикл поліморфного вірусу першого рівня
Авторська розробка

Поліморфні віруси другого рівня

Розшифровщик має постійною одну чи декілька команд, основна ж його частина непостійна. До такого типу поліморфних вірусів належать віруси, розшифровщики яких використовують альтернативні команди, різні регістри.

Розшифровщик вірусу має постійний алгоритм, але складається з довільно обраної послідовності команд. Розглянемо розшифровувач і підберемо синоніми для кожної його машинної команди:

MOV ESI,offset body_begin -> LEA ESI,body_begin;MOV ESI,offset body_begin-1/INC ESI;
MOV EDI, ESI -> PUSH ESI/POP EDI; XOR EDI,EDI/ADD EDI,ESI;
LODSB -> MOV AL,[ESI]/INC ESI; SUB AL,AL/SUB AL,[ESI]/NOT AL/INC ESI/ADD AL,1
XOR AL,66h -> XOR AL, 6/XOR AL, 60h

Для побудови кожної команди розшифровувача вибираються відповідні команди синоніми, тобто весь код розшифровувача складається з блоків команд синонімів. Результатом може бути наступний випадково згенерований розшифровщик:

```
LEA ESI, body_begin
    PUSH ESI
    POP EDI
    MOV ECX, offset body_end - body_begin + 2
    DEC ECX
    DEC ECX
my_begin:
    MOV AL,[ESI]
    INC ESI
    XOR AL, 6
    XOR AL, 66
    MOV [EDI], AL
    SUB EDI,-1
    ADD ECX,-1
    JNZ my_begin
```

Опишемо формально даний рівень поліморфізму за допомогою граматики.

Нехай є формальна граматика G2=(T, N, P, S), яка складається:

T={x,y – команди розшифровщика, x_1, $x_2, x_3, \ldots, x_n$; y_1 y_2 $y_3, \ldots, y_n$ – альтернативні команди розшифровщика };

N={R1,R2 - розшифровщика };

P – множина правил вибору розшифровщика;

S – початковий символ.

$$P = \begin{cases} S \to xR1 \mid yR2 \mid xS \mid yS \\ R1 \to x_1R1 \mid x_2R1 \mid x_3R1 \mid \ldots \mid x_nR1 \mid xS \mid \alpha \\ R2 \to y_1R2 \mid y_2R2 \mid y_3R2 \mid \ldots \mid y_nR2 \mid yS \mid \alpha \\ \alpha \to "" \end{cases}$$

Результатом може бути виведення:

$$S \to xS \to xyS \to xyxR1 \to xyxx_2R1 \to xyxx_2y_3R2 \to xyxx_2y_3$$

Життєвий цикл поліморфного вірусу другого рівня представлено на рисунку 3.

Рис. 3 .Життєвий цикл поліморфного вірусу другого рівня
Авторська розробка

Поліморфні віруси третього рівня

Поліморфні віруси такого рівня містять у своєму коді невикористовуванні команди (команди-сміття). Наприклад NOP; MOV AX,AX; CLI; CLD; STI та ін. Такий прийом дозволяє заплутувати власний код вірусу, а відповідно ускладнює процес виявлення такого вірусу.

Нехай генератор команд-сміття може створити чотири інструкції x86 – a,b,c,d. Тоді також необхідно згенерувати безпосередньо команди розшифровщика, назвемо їх командами корисного навантаження розшифровщика x та y. Тоді поліморфний генератор буде працювати так:

Нехай є формальна граматика G3=(T, N, P, S), яка складається[2,6]:

T={a,b,c,d – команди сміття, x, y – команди корисного навантаження шифрувальника/розшифровщика };

N={A,B} – нетермінальні символи;

P – множина правил вибору розшифровщика;

S – початковий символ.

$$P=\begin{cases} S\to aS\,|\,bS\,|\,cS\,|\,dS\,|\,xA \\ A\to aA\,|\,bA\,|\,cA\,|\,dA\,|\,yB \\ B\to aB\,|\,bB\,|\,cB\,|\,dB\,|\,\alpha \\ \alpha\to"" \end{cases}$$

Результатом виведення може бути:

$$S\to bS\to bcS\to bcdS\to bcdxA\to bcdxaA\to bcdxadA\to bcdxadyB\to bcdxadybB\to$$
$$\to bcdxadyb\alpha\to bcdxadyb$$

Генератор сміттєвих команд створює довільну кількість команд-сміття. Потім генерується xA, тобто команда, що несе корисне навантаження, а далі знову певна кількість команд-сміття.

Отже, поліморфний генератор є граматикою, що генерує команди відповідно до правил, що обмежені алфавітом. В залежності від того, наскільки повний алфавіт можна генерувати велику кількість слів[6]. Життєвий цикл поліморфного вірусу третього рівня представлено на рисунку 4.

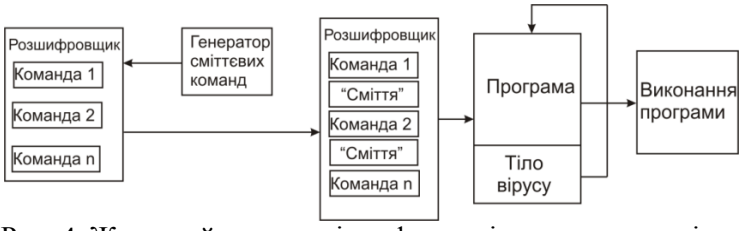

Рис. 4. Життєвий цикл поліморфного вірусу третього рівня
Авторська розробка

Поліморфні віруси четвертого рівня

В розшифровщику використовуються взаємозамінюванні команди та змінюється порядок їх слідування, тобто відбувається "перемішування" команд. Алгоритм розшифровки при цьому не

змінюється. Наприклад, команда MOV AX,BX може бути замінена такими еквівалентними командами: XCHG AX,BX; PUSH BX POP AX; MOV CX,BX MOV AX,CX;

В даному рівні поліморфізму також ускладнюється генераця сміттєвих інструкцій, робить їх менш очевидною. Наприклад, якщо ми бачимо MOV EAX, EBX, то перед цим у EAX можна писати все, що завгодно. Все одно регістр буде заново ініціалізований першим. Більш складним завданням є відслідковування звернень до регістрів, виявлення невикористовуваних регістри (як локально, так і глобально), і заповнення його сміттям. Для цього потрібен не тільки дизассемблер довжин, але й повноцінний дизассемблер команд, що визначає, що це саме MOV EAX, EBX, а не щось інше. Причому, необхідно спеціальним чином обробляти прапори - зустрівши команду, залежну від прапорів (наприклад, Jxx), необхідно знайти найближчу до неї інструкцію, що впливає на цей прапор і між ними вставляти тільки ті команди-сміття, яке не чинить на прапори ніякого впливу.

Опишемо даний рівень за допомогою формальної граматики.

Нехай є формальна граматика G4=(T, N, P, S), яка складається:

T={a,b – команди-сміття, x,y,z – команди корисного навантаження шифрувальника/розшифровщика};

N={A,B,C} – нетермінальні символи;

P – множина правил вибору розшифровщика;

S – початковий символ.

$$P = \begin{cases} S \to yB \mid aB \mid C \\ C \to zC \mid aC \mid A \\ A \to xA \mid aA \mid S \mid \alpha \\ \alpha \to "" \end{cases}$$

Ланцюжок виведення може виглядати:

$$S \to yB \to yaB \to yaC \to yazC \to yazA \to yazxA \to yazx\alpha \to yazx$$

Життєвий цикл поліморфного вірусу четвертого рівня представлено на рисунку 5.

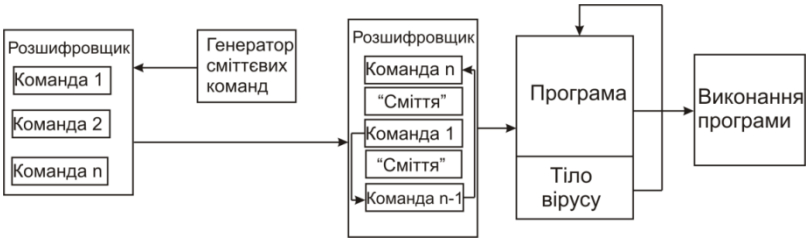

Рис.5 . Життєвий цикл поліморфного вірусу четвертого рівня
Авторська розробка

Поліморфні віруси п'ятого рівня

Включає в себе використання всіх перерахованих вище рівнів. Розшифровувач може використовувати різні алгоритми розшифрування вірусного коду. Також можливе використання, для розшифрування основного вірусного коду, розшифровки частини самого розшифровщика або декількох розшифровщиків, що почергово розшифровують один одного, або, безпосередньо, вірусний код. Як правило, детектування вірусів даного рівня поліморфізму за допомогою сигнатури неможливо. Процес детектування і, особливо, лікування такого вірусу дуже складний і може бути досить тривалим за часом(difficult and time-consuming task). Якщо для детектування такого вірусу можливий серйозний аналіз коду тільки самого розшифровщика, то для лікування необхідно зробити часткову або повну розшифровку тіла вірусу, для отримання оригінальної інформації про заражений файл.

Нехай є формальна граматика G5=(T, N, P, S), яка складається:

T={G1,G2,G3,G4 – формальні граматики (описані вище);

N={A,B,C,D} – множина нетермінальних символів ;

P – множина правил вибору розшифровщика;

S – початковий символ.

$$P = \begin{cases} S \to A \mid B \mid C \mid D \\ A \to G1 \mid \alpha \\ B \to G2 \mid \alpha \\ C \to G3 \mid \alpha \\ D \to G4 \mid \alpha \end{cases}$$

$$((G1,G2,G3,G4)^* \in G5) \in UL$$

Життєвий цикл поліморфного вірусу п'ятого рівня представлено на рисунку 6.

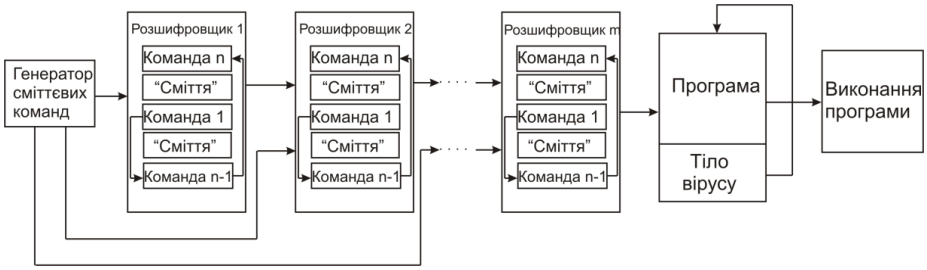

Рис. 6. Життєвий цикл поліморфного вірусу п'ятого рівня
Авторська розробка

Поліморфні віруси шостого рівня

Такі поліморфні віруси складаються з програмних блоків. Вони постійно змінюються в тілі і переміщують свої підпрограми (інсталяції, зараження, обробника переривання, аналізу файлу і т.д.). Характерною особливістю таких вірусів є наявність "плям". При цьому в різні місця файлу записується кілька блоків коду, що обумовлює назву методу. Такі плями в цілому утворюють поліморфний розшифровщик, який працює з кодом в кінці файлу. Для реалізації методу навіть не потрібно використовувати команди-сміття - підібрати сигнатуру буде неможливо. Подібні віруси можуть бути незашифровані.

Весь оброблюваний код ділиться на блоки постійного або змінного розміру, які в кожному поколінні вірусу переставляються у випадковому порядку. Тим не менш, при програмуванні виникають наступні проблеми - оскільки адреси блоків в кожному поколінні міняються, машинний код повинен бути повністю переміщуваним, тобто зберігати працездатність незалежно від свого місця розташування. Це досягається шляхом відмови від безпосередніх міжблочних викликів. Здійснювати переходи, викликати функції, звертатися до змінних можна тільки в межах "свого" блоку. У практичному плані це означає, що разом з кодом кожен блок несе й свої змінні. Але все-таки робити міжблочні виклики іноді доводиться. Найпростіше створити таблицю з базовими адресами всіх блоків і розмістити її по фіксованому зміщенню - наприклад, покласти в перший блок.

Нехай є формальна граматика G5=(T, N, P, S), яка складається:

T={a,b,c – команди блоку A,B,C відповідно};

N={A,B,C} – не термінальні символи;

P – множина правил вибору розшифровщика;

S – початковий символ.

$$P = \begin{cases} S \to A \mid B \mid C \mid \alpha \\ A \to aA \mid B \mid C \\ B \to bB \mid A \mid C \\ C \to cC \mid A \mid B \end{cases}$$

Життєвий цикл поліморфного вірусу шостого рівня представлено на рисунку 7.

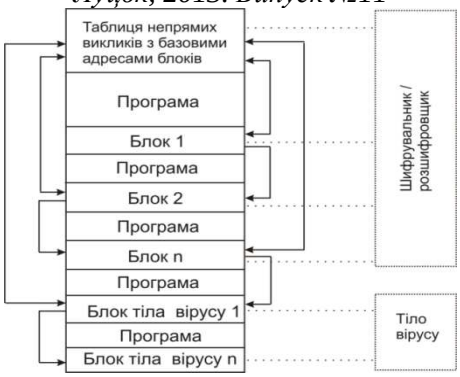

Рис. 6.Життєвий цикл поліморфного вірусу шостого рівня
Авторська розробка

Висновок

В ході дослідження було проаналізовано класифікацію поліморфних вірусів. На основі даної класифікації розроблені моделі життєвого циклу поліморфних вірусів, що описуються за допомогою формальної граматики. Описані моделі життєвого циклу поліморфних вірусів створюють простір для побудови методу діагностування комп'ютерних систем на наявність поліморфного шкідливого коду. Наприклад, весь вірусний код може бути інтерпретований як ланцюжок граматики G3, де кожен символ ланцюжка є певною командою x86. Тому, наступним етапом є розробка методу, що полягає у розборі таких ланцюжків та аналізу належності кожного символу до ланцюжка, що відноситься до відповідної граматики.

1. Szor, P. Hunting for Metamorphic / P. Szor, P. Ferrie// Virus Bulletin. -Sept, 2001. -P. 123-144.
2. П.В. Збицкий. Модель метаморфного преобразования исполняемого кода / П.В. Збицкий // Серия "Компьютерные технологии, управление, радиоэлектроника", выпуск 10, 2009.-С. 57-61.
3. Гордон Ян. Компьютерные вирусы без секретов. – М.: Новый издательский дом, 2004, - С. 138-142
4. Understanding and Managing Polymorphic Viruses, 1996, Symantec, The Symantec Enterprise Papers, vol. XXX, pp. 1-13. cited by other.
5. Крис Касперски. Ступени полиморфизма / Электронный журнал Xaker 10/05 // режим доступу [http://www.xakep.ru/magazine/xa/082/120/1.asp], дата звернення 10.04.2013
6. Qozah. Polymorphism and grammars / Qozah// 29A E-zine. - 1999. - № 4.
7. Bruschi, D. Using Code Normalization for Fighting Self-Mutating Malware / D. Bruschi, L. Martignoni, M. Monga // Security & Privacy, IEEE. -2007. -V. 5. -P. 46-54.
8. Гладкий, А.В. Формальные грамматики и языки / А.В. Гладкий. —М.: Наука, 1973. - 368 с.
9. Електронний журнал Xaknotdie, VMag, Issue 3, 1 January 1999 / И.Данилов / [Електронний ресурс] / Режим доступу http://xaknotdie.org/lamersmustdie3/VIRII/POLY_ID.html (дата звернення 15.03.2013)