

УДК 004.056.54

О. С. САВЕНКО

Хмельницький національний університет

## ФОРМАЛІЗОВАНЕ СТРУКТУРУВАННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АЛГЕБРАЇЧНИХ СИСТЕМ

*В роботі представлено розроблені алгебраїчні системи та алгебри з введеними операціями на множині шкідливого програмного забезпечення. Вони є основою для створення поведінкових сигнатур шкідливого програмного забезпечення з метою їх формалізованого представлення в системах виявлення. Особливістю розроблених алгебраїчних систем є поділ та структуризація шкідливого програмного забезпечення за типами. Це дозволило здійснювати їх розподіл і віднесення до підмножин на основі характеристичних властивостей шкідливого програмного забезпечення для проведення подальшої ідентифікації та класифікації.*

*Ключові слова:* шкідливе програмне забезпечення, алгебраїчні системи, алгебри, поведінкові сигнатури.

O. SAVENKO

Khmelnitsky National University

## FORMALIZED STRUCTURING OF MALICIOUS SOFTWARE BASED ON ALGEBRAIC SYSTEMS

*The paper presents developed algebraic systems and algebras with introduced operations on a set of harmful software. They are the basis for creating behavioral signatures of malicious software in order to formalize their representation in detection systems. The peculiarity of the developed algebraic systems is the division and structuring of malicious software by types. This allowed to distinguish their distribution and assignment to subsets on the basis of the characteristic properties of malicious software for further identification and classification. The use of such a representation of the SZR will require the development of methods for detecting it by elaborating the characteristics of the characteristic behaviors, which, respectively, will attribute the elements of the SPS to certain subsets. The elaborated algebraic systems and algebras with the introduced operations on the SZP set are the basis for the creation of behavioral signatures of SPSs with the aim of their formalized representation in detection systems. The peculiarity of the developed algebraic systems is the division and structuring of the SFR by types, which allows them to be distributed and assigned to subsets based on the characteristic properties of the SPS for identification and classification.*

*Keywords:* malicious software, algebraic systems, algebras, behavioral signatures.

ВСТУП. Засоби виявлення шкідливого програмного забезпечення (ШПЗ) на сьогодні не задовільняють потреб користувачів. Переважно виявлення ШПЗ відбувається вже після того, коли воно поширювалось на протязі певного часу і виконувало деструктивні дії. Відомі різноманітні антивірусні засоби, які здійснюють виявлення ШПЗ на різних етапах його життєвого циклу, не забезпечують високої достовірності його виявлення. ШПЗ побудоване, як складні багатофункційні програмні системи з використанням ефективних методів створення програмних засобів та методів поширення зловмисного коду. Для організації ефективної протидії таким засобам, необхідним є подальший розвиток теорії та практики створення систем виявлення ШПЗ. Окремою проблемою, яка потребує дослідження та вирішення, є представлення об'єктів ШПЗ такими формальними структурами, що дозволили б підвищувати достовірність його ідентифікації. На сьогодні недостатньо описано та формалізовано ШПЗ, як об'єкти для ідентифікації, а також як певну загальноприйнятну формалізовану систему, яка була б стала основою для подальшого розвитку.

Розглядатимемо здійснення функціонування шкідливого програмного забезпечення в розподілених обчислювальних системах комп'ютерних мереж.

ПОВ'ЯЗАНІ РОБОТИ. Перше означення та модель комп'ютерного вірусу представив Ф. Коен. Зокрема в [1] на основі машини Тюрінга були описані множини комп'ютерних вірусних програм через таку їх особливість, як поширення вірусного коду в плоскій моделі пам'яті. При такому переміщенні отримуються нові елементи, які здатні знову поширювати себе. Розроблена модель не враховує конструкторних особливостей комп'ютерних систем і зокрема їх розміщення в мережах.

Модель Л. Адлемана, яка представлена в [2] базується на нумеруванні символів та призначенні їм Геделевого номеру, за яким шляхом певних обчислень можна було б встановити приналежність програми до множини ШПЗ. Розроблена ним модель була використана для встановлення можливості спроможності розрізнити комп'ютерних вірус та корисну програму і для оцінки обчислювальної складності цього процесу.

В роботі [3] Ж. Бонфана, М. Качмарекка і Ж.-І. Маріона запропонована модель визначення комп'ютерних вірусів, яка базується на теоремі Кліні з теорії алгоритмів, згідно якої, по аналогії з моделлю Л.Адлемана, кожній функції ставиться у відповідність число. Тоді, результатом функції від числа буде саме це число, що означає можливість тиражування програм. Запропонована модель була використана для більш точної оцінки обчислення складності алгоритмів розпізнавання комп'ютерних вірусів.

Подловченко Р.І. в роботі [4] було запропоновано формальні моделі програм та використано їх для оцінки складності обчислювального процесу.

Тому, відомі моделі дозволяють вирішити оцінку складності обчислювального процесу при визначенні віднесення програм до множини ШПЗ, надають формальне представлення означення комп'ютерних вірусів, але не в повній мірі охоплюють все ШПЗ і не можуть бути основою для практичної інтерпретації всього ШПЗ з метою його представлення для підвищення достовірності ідентифікації.

**ПОСТАНОВКА ПРОБЛЕМИ.** Таким чином, актуальною науковою проблемою є опис та формалізація ШПЗ, як об'єктів для ідентифікації, та розробка формалізованої системи для представлення знань про ШПЗ, яка б використовувалась при розробці засобів виявлення ШПЗ та стала б основою для подальшого розвитку.

**ОСНОВНА ЧАСТИНА.** Введемо позначення для множини всього шкідливого програмного забезпечення  $V$ , яке перебуває в комп'ютерних системах локальних мережах. Тобто розглядатимемо те ШПЗ, яке за певних обставин та на протязі певного часу експлуатації локальних комп'ютерних мереж, проникло в комп'ютерні системи, змогло пройти певні системи захисту і функціонує там. Представимо ШПЗ в локальних комп'ютерних мережах, особливістю якого є втілення у виконувани файли, завантажувальний сектор жорсткого диску, оперативний запам'ятовуючий пристрій та поширення мережею своїх копій, алгебраїчною системою типу  $\tau = (\alpha, \beta)$ :

$$\mathfrak{A}_V = \langle V, \Omega_F, \Omega_P \rangle, \quad (1)$$

де  $\Omega_F = \{F_0, F_1, F_2, \dots, F_{\alpha_1}, \dots\}$  – множина операцій заданих на множині  $V$  для кожного  $\alpha_1 = 0, 1, 2, \dots$ ;  $\Omega_P = \{P_0, P_1, P_2, \dots, P_{\beta_1}, \dots\}$  – множина предикатів заданих на множині  $V$  для кожного  $\beta_1 = 0, 1, 2, \dots$ ;  $\alpha = 1, \beta = 1$  – арності операцій, тому тип системи  $\tau = (1, 1)$ . Елементами множини  $v_j \in V$  ( $j = 1, 2, \dots$ ) вважатимемо всі об'єкти файлової системи, завантажувального сектору диску, оперативної пам'яті, мережні пакети, які відносяться до розглядуваного ШПЗ. Елементи  $v_0 \in V_0 \subseteq V$  є одиничними елементами, тобто такими що містять єдиний функціонал, вміст якого полягає у необхідності здійснення самокопіювання з метою поширення, але без конкретного функціонального наповнення для виконання технічно цих дій. Решта операцій представлені іншими функціями. Ці елементи, що формують множину  $V_0$  є породжуючими для решти різних елементів множини  $V$ . Функції з множини  $\Omega_F$  виконуються на елементах  $v_0$ , що формує інші об'єкти, які належатимуть множині  $V$ , а також можуть виконуватись на інших елементах множини  $V$ , які не належать множині  $V_0$ . Функції з множини  $\Omega_P$  не завжди успішно виконуватимуться по відношенню до елементів з множини  $V$ , тому для представлення ШПЗ в локальних мережах вибрано також множину предикатів, яка відображатиме результат успішного/неуспішного виконання функцій.

Функції  $F_{\alpha_1}$  ( $\alpha_1 = 0, 1, 2, \dots$ ) з множини  $\Omega_F$  визначимо, як такі що здійснюватимуть відображення елементів з множини  $V$  на неї. Їх конкретне визначення залежатиме від поділу множини  $\Omega_F$  на підмножини за різними характеристичними властивостями ШПЗ. Предикати  $P_{\beta_1}$  ( $\beta_1 = 0, 1, 2, \dots$ ) з множини  $\Omega_P$  визначимо, як такі що будуть істинними при успішному виконанні операцій і хибними – в іншому випадку.

Множину  $\Omega_F$  представимо її підмножинами  $\Omega_{F_{\alpha_1}}$ , які відображатимуть такі характеристичні для ШПЗ властивості та закладені в його функціонал особливості:

- 1) зберігання знань про механізм місцерозміщення своїх наступних копій;
- 2) пошук місця в пам'яті для розміщення своєї копії;
- 3) знання про механізми втілення у виконувани програми;
- 4) механізми запису в оперативну пам'ять;
- 5) приховування свого перебування в комп'ютерних системах;
- 6) пошук інших вузлів мережі для свого поширення;
- 7) механізми для формування і відправки мережних пакетів;
- 8) подолання механізмів захисту;
- 9) техніки запису своїх копій в головний завантажувальний сектор;
- 10) виконання деструктивних дій.

Ці характеристичні властивості ШПЗ пов'язані з системними викликами, які відносяться до роботи з файлами, оперативною пам'яттю та командами роботи в мережі: створення, відкриття, закриття, видалення, читання, записування, додавання, знаходження, отримання атрибутів і встановлення атрибутів, команди доступу до ОП, команди для роботи в мережі.

Використання розроблених алгебраїчних систем та алгебр надасть змогу здійснити узагальнення та формалізацію предметної області для структурування ШПЗ за правилами його поведінки. Ці алгебраїчні структури міститимуть наповнення конкретними знаннями функціонування ШПЗ в процесі його життєвого циклу і будуть основою для розробки методів їх виявлення.

Для відображення різних особливостей ШПЗ, які проявляються в його різних типах, представимо множину ШПЗ  $V$  підмножинами  $V_l \subseteq V$ , де  $l = \overline{1, w}$ ,  $V = \bigcup_{l=1}^w V_l$ , де  $w$  – кількість типових поділів

шкідливого програмного забезпечення за певними ознаками чи критеріями. Все шкідливе програмне забезпечення, яке може функціонувати в локальних комп'ютерних мережах, можна реалізувати за допомогою базових операторів, які допускаються для виконання конкретного комп'ютерною системою, і їх множину позначимо через  $P_1$ , та базових предикатів, які відповідають елементарним заданим відношенням між даними програми. Базовими операторами вважатимемо виклики бібліотечних функцій, базовий елемент програми або автономний фрагмент програми. Базові предикати приймають значення істинне або хибне.

Множину всіх підмножин  $P_r$  позначимо через  $\mathfrak{P}(P_r)$  і тоді елементи з цієї множини відповідають всім можливим значенням базових предикатів. Важливим при цьому представленні програми через базові оператори та предикати є те, що для створення програми одні і ті ж базові оператори можуть використовуватись багаторазово, а не одноразово. Крім того, до шкідливих програм також віднесемо завершені фрагменти програм, які призначені для виконання допоміжних дій при підготовці зловмисних дій. Базові оператори та предикати, які розглядатимемо при подальших випадках, вважатимемо співвіднесеними до мови програмування найбільш наближеної до апаратури комп'ютерної системи або обладнання комп'ютерної мережі.

Графічне представлення множини ШПЗ, множини базових операторів та предикатів і співвідношення між ними, на прикладі однієї програми зображено на рис.1.

Елементи мови програмування позначимо через  $p_j$ ,  $j=1, m$ ,  $p_j \in P$ ,  $P = P_1 \cup P_2$ . Елементи множини ШПЗ позначимо з врахуванням їх приналежності конкретній підмножині так:

$$\begin{aligned} V_1 &= \{V_{11}, V_{12}, \dots, V_{1n_1}\}; \\ V_2 &= \{V_{21}, V_{22}, \dots, V_{2n_2}\}; \\ &\dots \\ V_n &= \{V_{n1}, V_{n2}, \dots, V_{nn_k}\}, \end{aligned} \quad (2)$$

де  $n_j$  – кількість елементів ШПЗ  $j$ -го класу,  $j=1, n_k$ .

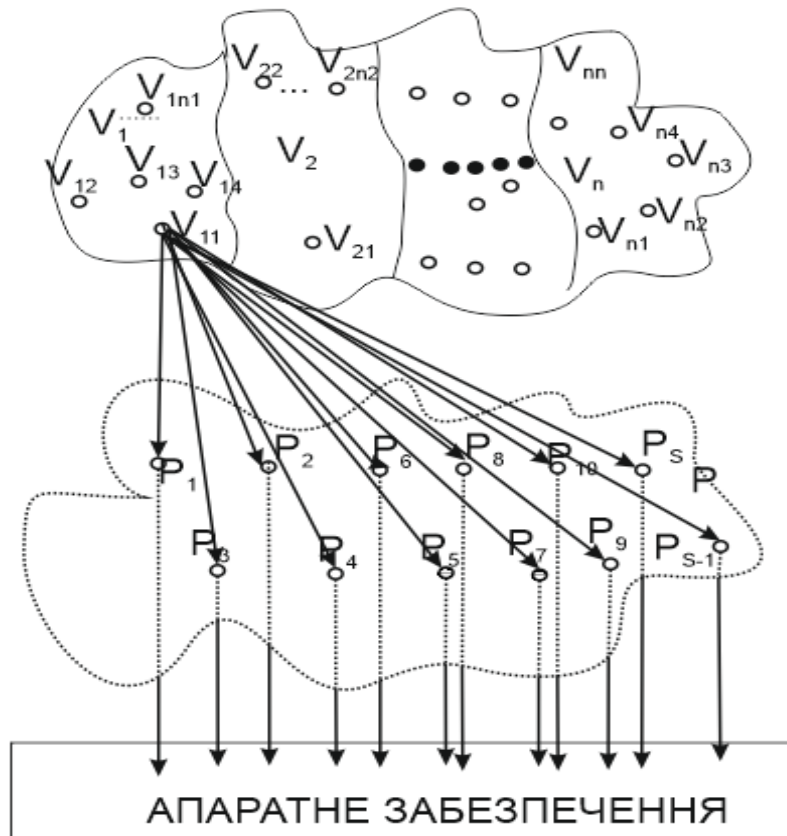


Рис. 1. Представлення елементів множини V

Для конкретної шкідливої програми  $v_{ij}$  зображеної на рис. 1 зв'язками з елементами множини P іншого ШПЗ та аналогічних елементів з V для мови кількість та послідовність використаних елементів з P можна представити в табл. 1

Таблиця 1

Кількість входжень елементів множини P					
$V_{11}$	$\alpha_1$	$\alpha_2$	$\alpha_2$	.....	$\alpha_s$
	$p_1$	$p_2$	$p_3$	.....	$p_s$

де  $\alpha_k$ ,  $k = \overline{1, s}$  – кількість входжень елементу  $p_k$  в структуру  $v_{ij}$ , якщо враховувати збереження послідовності, тоді представлення буде у вигляді строгої послідовності. Як в першому, так і в другому, поданні елементу множини V можна виділити унікальну послідовність, як правило представлену байтами, за якою можна ідентифікувати елемент ШПЗ фактично встановлюючи відповідність шаблону, якщо таке ШПЗ вже відоме і з нього отримано шаблон. Використання такого шаблону можливе на практиці, якщо він є незмінним при кожному тиражуванні (розмноженні) елементу ШПЗ. Якщо ж при розповсюдженні ШПЗ змінює свій код, тоді використання такого шаблону неефективне.

Кількість елементів  $p_i$ ,  $i = \overline{1, s}$  множини P скінченна і враховуючи особливості архітектури сучасних процесорів є не дуже великою. Але через можливість багатократного використання одних і тих же елементів  $p_j$ , тобто коефіцієнти  $\alpha_j$ ,  $j = \overline{1, s}$  можуть бути великими, то представлення елементів  $v_{ij} \in V$  може бути дуже різноманітним і як наслідок зводиться до комбінації з повтореннями:

$$P_{\alpha'}(\alpha_1, \alpha_2, \dots, \alpha_s) = \frac{\alpha'!}{\alpha_1! \alpha_2! \dots \alpha_s!}, \quad (3)$$

де  $\alpha' = \alpha_1 + \alpha_2 + \dots + \alpha_s$ . На оцінку можливого результату значення  $P_{\alpha'}$  впливатимуть величини  $\alpha_j$ , які можуть бути великими, але скінченними і як наслідок:

$$\left| \lim_{\substack{\alpha_j \rightarrow q \\ 1 \leq j \leq s}} \frac{(\alpha_1 + \alpha_2 + \dots + \alpha_s)!}{\alpha_1! \alpha_2! \dots \alpha_s!} \right| \leq \frac{(s \times q)!}{s \times q!}, \quad (4)$$

де  $q = \max\{\alpha_j\}$ ,  $j = \overline{1, s}$ .

Як правило, елементи ШПЗ прагнуть створювати з використанням мінімізованої кількості команд. Тому, величина q вважається невеликою. Але використання сучасних середовищ програмування та урізноманітнення засобів приховування шкідливого коду призводить до суттєвого збільшення величини q і як наслідок величини  $P_L$ . Це суттєво впливає на збільшення елементів унікального шаблону та відповідно на час пошуку через ускладнення обчислювального процесу.

Бази сигнатур шкідливих програм (шаблонів), які сформовані в сучасних антивірусних інформаційних технологіях поповнювалися більше 20 років і містять досить великі об'єми даних. Створення нових антивірусних інформаційних технологій на основі сигнатурного методу не дозволить конкурувати з тими, які вже накопичили свої бази сигнатур, а враховуючи зняття обмежень на величину q стає неперспективним. Сигнатурний метод можна віднести до точних методів, якщо розглядати його як задачу знаходження повного співпадіння шаблону.

В зв'язку з неможливістю поповнення бази сигнатур шаблонами відомих вірусів, розробка нових систем виявлення ШПЗ на основі такого методу не перспективна, тому актуальним напрямком дослідження є розробка нових та вдосконалення існуючих моделей, методів та систем виявлення ШПЗ, які дозволяють здійснювати пошук нового ШПЗ не за їх сигнатурами, а використанням бази сигнатур було б допоміжним засобом.

Формалізуємо область множини P та відношення між її величинами за допомогою алгебраїчних систем. Введемо операцію об'єднання ( $\cup$ ) на множині  $\mathfrak{P}(P)$  і будемо розглядати її як об'єднання всеможливих підмножин множини P:

$$\mathfrak{S}_P = \langle \mathfrak{P}(P); \cup; \subseteq \rangle. \quad (5)$$

Дійсно, елементами множини  $\mathfrak{P}(P)$  є всеможливі підмножини множини P, тоді їх об'єднання утворює множину, в якій елементи, що повторюються замінюються в цій множині один раз і тоді утворений перелік фактично буде однією з підмножин множини  $\mathfrak{P}(P)$ . В якості нулевого елементу виступає порожня множина.

Операція включення ( $\subseteq$ ) відповідатиме за те, чи є серед затребуваних до виконання інструкцій всі елементи з P чи ні. Елемент ШПЗ може включати порожню множину. Може виявитись, що серед елементів множини V є такі, що в їх структурі використані елементи, яких немає в P.

Алгебраїчна структура  $\mathfrak{S}_P$  має тип <2.2>, бо операції, задані на множині  $\mathfrak{P}(P)$  є двомісними. Операції  $\cup$  та  $\subseteq$  є головними. Множина функцій  $\Sigma_{F_T} = \{\cup\}$ , а множина предикатів  $\Omega_{\mathfrak{P}(P)_T} = \{\subseteq\}$ .

Виходячи з алгебраїчної структури  $\mathfrak{S}$  задано алгебру компонентів шкідливих програми  $\mathfrak{B}_T = \langle \mathfrak{P}(P); \cup \rangle$  та модель  $\mathfrak{M}_T = \langle \mathfrak{P}(P); \subseteq \rangle$ .

Потужність множини  $\mathfrak{P}(P)$  позначимо  $|\mathfrak{P}(P)|$  і є для алгебраїчної системи  $\mathfrak{B}_T$  її порядком. Потужність  $\mathfrak{P}(P)$  є скінченною, так як множина  $P$  скінчена. Це обмеження є необхідною умовою для доцільності побудови моделей пошуку ШПЗ за критерієм ефективності. Дійсно, якщо б множина  $P$  була нескінченною і ця її нескінченність досягалась би за рахунок композиції елементів, тоді вона була злічена, і як наслідок потужність розглядуваної множини  $\mathfrak{P}(P)$  породженої множиною  $P$  мала потужність континууму.

Введення таких алгебраїчних структур дозволяє формалізувати простір, в якому перебувають розглядувані елементи з множини ШПЗ, і є основою для дослідження різних типів ШПЗ шляхом додавання нових операцій над елементами множини  $\mathfrak{P}(P)$ . Представимо елементи  $v_{ij} \in V_i, V_i \subseteq V_j$  для  $i=\overline{1, n}, j=\overline{1, n_1}$  через елементи множини  $\mathfrak{P}(P)$  наступним набором:  $v_{ij} = (\{p_1\}; \{p_1, p_2\}; \dots)$ .

Множина  $V$  формується з елементів, які обов'язково мають хоча б одну з ознак – властивостей, що відносяться до ШПЗ. За цими властивостями, як за відношенням, множина  $V$  поділяється на класи – підмножини  $V_i, i = \overline{1, n}$ . Всі елементи множини  $V$  входять до множини всіх програм, але за виділеними ознаками вони формують лише множину  $V$ , яка є підмножиною множини всіх програм. До складу множини  $V$  за виділеними ознаками можуть відноситись і не тільки елементи множини ШПЗ, які мають згідно з вимогами до свого функціоналу функції, які є серед виділених ознак. Якщо  $S$  множина ознак – властивостей, тоді  $\mathfrak{P}(S)$  - множина підмножин множини  $S$  і  $V = \bigcup \mathfrak{P}(S)$ , тому можна ввести алгебраїчну систему так:

$$\mathfrak{S}_V = \langle V; \cup; \subseteq \rangle, \quad (6)$$

де  $\cup$  – операція об'єднання, яка задана на множині  $V$ ;  $\subseteq$  – операція включення, тобто якщо програма має одну з ознак-властивостей, тоді вона означає включення до множини  $V$  всіх програм, в яких є ознака з множини  $S$ . Для достовірної класифікації програми з множини  $V$  необхідні методи, які б враховували ці ознаки з  $S$ . Є частина корисних програм (наприклад, архіватори), які мають подібні властивості і можуть бути помилково віднесеними в множину  $V$ .

Розглянемо множину  $V_1$ , в яку входять всі елементи ШПЗ з ознаками-властивостями і задамо алгебраїчну систему так:

$$\mathfrak{S}_{V_1} = \langle V_1; F_1; P_1 \rangle, \quad (7)$$

функцію  $F_1: V_1 \rightarrow P$ ; предикат  $P_1$  заданий так:

$$F_1(v_{1i}, p_j) = \begin{cases} 0, \text{ не наявний } p_j \text{ в } v_{1i}, \\ 1, p_j \text{ міститься в } v_{1i} \end{cases} \quad (8)$$

Функція  $F_1$  може бути задана декількома способами. Наприклад:

$$\begin{array}{ccc} F_{11}: & \{P_1\} & \alpha_{11} \\ & \{P_2\} & \alpha_{21} \\ & \dots & \dots \\ & \{P_n\} & \alpha_{n1} \\ & \{P_1, P_2\} & \rightarrow \alpha_{121}, \alpha_{221} \\ & \{P_1, P_3\} & \alpha_{122}, \alpha_{322} \\ & \dots & \dots \\ & \{P_1, P_2, P_3\} & \alpha_{131}, \alpha_{23}, \alpha_{33} \\ & \dots & \dots \end{array}$$

Тобто  $F_{11}(v_{1j}) = (\alpha_{11}; \alpha_{21}; \dots; \alpha_{n1}; \alpha_{121}; \alpha_{221}; \alpha_{122}; \alpha_{322}; \dots; \alpha_{131}; \alpha_{23}; \alpha_{33}; \dots)$ , що означає представлення послідовного входження кожного елементу з  $\mathfrak{P}(P)$  у  $v_{1j} \in V_1, i=\overline{1, n_1}$  числом. Для встановлення взаємно – однозначної відповідності елементи з  $P$  попередньо строго впорядковуються і тоді в подальшому кожен елемент з  $v_{1j}$  порівнюється з ними. В результаті за числами  $\alpha_{i_1(i_2, i_3, i_4, \dots, i_{2n_1})}$  можна відтворити  $v_{1j}$ . Взявши фрагмент або фрагменти такої числової послідовності отримаємо сигнатуру, представлену в числовому вигляді.

Друга функція може не включати кількість входжень  $p_i, i = \overline{1, s}$ , а ставитиме у відповідність  $v_{1j}$  відповідні елементи  $j$  безпосередньо:

$$F_1 = (v_{1i}) = (p_{i_1}, p_{i_2}, \dots, p_{i_t}), \text{ де } t - \text{кількість елементів у } V_{1i}.$$



Третій варіант можна побудувати так:

$$F_{13}: v_{1i} \rightarrow p_1, \alpha_1 \\ p_2, \alpha_2 \\ \dots \\ p_s, \alpha_s.$$

$F_{13}(v_{1i}) = ((p_1; \alpha_1), (p_2; \alpha_2), \dots, (p_s; \alpha_s))$ , тобто визначити кількість входжень кожного з елементів  $p_j$  в  $v_{1i} \in V$ .

Функції  $F_{11}, F_{12}$  однозначно визначають елементи, з яких складаються  $v_{1i} \in V_1$ .

Функція  $F_{13}$  не завжди є взаємно однозначною, особливо у випадках, коли кількість елементів  $p_i$  в  $v_{1i} \in V_1$  є мінімальною, тоді входження  $p_i$  можуть бути однаковими для різних  $v_{1i} \in V_1$ , а також і для корисних програм.

Елементи  $v_{1i} \in V_1$  можуть мати однакові властивості, але бути представлені різними наборами  $p_j \in P$ . Тоді поділимо множини  $V_i$  на підмножини за ознакою чи групою ознак. Тобто на множинах  $V_i$  можна ввести відношення еквівалентності за ознаками – властивостями. Якщо розглядати множину  $V$  з введеним на ній відношенням еквівалентності, то класи еквівалентності формуватимуть множини  $V_i$ ,  $i=1, n, \bigcap_{i=1}^n V_i = \emptyset$ .

На практиці для знаходження  $V_i$ ,  $i=1, n$  таких, що їх перетин є порожньою множиною, тобто однозначно віднести елементи  $v_{ij}$  до певної множини  $V_i$  не завжди є вирішуваною задачею і тому відношення еквівалентності потрібно вводити за умови укрупнення ознак – властивостей.

Для здійснення ідентифікації та кластеризації введемо відношення еквівалентності на кожній з множин  $V_i$ ,  $i = 1, n$ . Представимо їх наступними алгебрами:

$$\mathfrak{B}_S = \langle V, \cup, \leftrightarrow \rangle, \quad (9)$$

$$\mathfrak{B}_{V_i} = \langle V_i, \cup, \leftrightarrow \rangle. \quad (10)$$

Використання такого представлення ШПЗ потребуватиме при розробці методів його виявлення деталізації особливостей характерних поведінок, які відповідно відноситимуть елементи ШПЗ до певних підмножин.

**ВИСНОВКИ.** Розроблені алгебраїчні системи та алгебри з введеними операціями на множині ШПЗ є основою для створення поведінкових сигнатур ШПЗ з метою їх формалізованого представлення в системах виявлення. Особливістю розроблених алгебраїчних систем є поділ та структуризація ШПЗ за типами, що дозволяє здійснювати їх розподіл і віднесення до підмножин на основі характеристичних властивостей ШПЗ для проведення ідентифікації та класифікації.

#### Література

1. Cohen F. Computational aspects of computer viruses / F. Cohen // Computers and Security. – 1989. – Vol. 8. – P. 325–344.
2. Adleman L. An Abstract Theory of Computer Viruses / L. Adleman // CRYPTO '88. – P. 354–374.
3. Bonfante G. A Classification of viruses through recursion theorems / G. Bonfante, M. Kaczmarek, J.-Y. Marion // CiE, 2007. – P. 73–82.
4. Подловченко Р.И. Регулярные модели программ / Р.И. Подловченко, Н.А. Аланакян // Программирование. – 1993. – № 4. – С. 3–11.

#### References

1. Cohen F. Computational aspects of computer viruses / F. Cohen // Computers and Security. – 1989. – Vol. 8. – P. 325–344.
2. Adleman L. An Abstract Theory of Computer Viruses / L. Adleman // CRYPTO '88. – P. 354–374.
3. Bonfante G. A Classification of viruses through recursion theorems / G. Bonfante, M. Kaczmarek, J.-Y. Marion // CiE, 2007. – P. 73–82.
4. Podlovchenko R.I. Reguljarnyie modeli programm / R.I. Podlovchenko, N.A. Alanakyan // Programmirovanie. – 1993. – № 4. – S. 3–11.