УДК 658.512.011: 681.326: 519.713

СИГНАТУРНО-КУБІТНІ МЕТОДИ РОЗПІЗНАВАННЯ ДЕСТРУКТИВНИХ КОДІВ

АДАМОВ О.С., ХАХАНОВ В.І.

Пропонуються унітарно кодовані кубітно-матричні моделі, структури даних, обчислювальні архітектури і методи паралельного логічного аналізу деструктивних кодів у кіберфізичному просторі. Вводяться кубітні векторні структури даних для опису параметрів змінних, що беруть участь у формуванні еталонних зразків (паттернів) деструктивних вихідних кодів. Пропонується паралельний сигнатурно-кубітний метод моделювання malware даних для визначення їх належності до існуючих деструктивних компонентів malware library. Пропонується сигнатурно-кубітний метод синтезу еталонних логічних схем malware-функціональностей, який відрізняється від аналогів унітарним кодуванням сигнатур для кодів деструктивних компонентів і формуванням кубітних матриць. Вводиться сигнатурно-кубітний процесор активного online кіберфізичного cyber security комп'ютинга (CSC) на основі моніторингу вхідних malware-даних і їх моделювання на еталонних логічних схемах malware-функціональностей з метою подальшого актюаторного управління процесом видалення деструктивних компонентів.

Ключові слова: кубітно-матричні моделі, кіберфізичний простір, обчислювальні архітектури, паралельний логічний аналіз, деструктивний код.

Key words: qubit-matrix models, cyber-physics space, computational architecture, parallel logical analysis, malware.

1. Визначення та правила CSC-комп'ютингу

Розвиток кіберпростору перетворює фізичний світ з домінуючого у підлеглий. Реальний світ все більш залежить від віртуального, бо керується ним, оскільки всі фізичні процеси і явища мають власні цифрові образи, які поступово трансформуються в прообрази. Хто є лідером у кіберпросторі, той керує фізичним світом. Кіберфізичний світ позитивно з'єднує всіх жителів планети один з одним без посередників, завдяки соціальним мережам, хмарним сервісам і Edge Computing. Тема Cyber Security Computing інтегрально має хороший індекс в бібліотеках IEEE Xplore – 10980 і Springer – 23345. Проте існують поодинокі публікації в частині активного комп'ютингу, спрямованого на автоматичне виявлення і усунення malware без участі людини [1, 2]. Природно, що з'єднання двох ринково-орієнтованих наукових напрямків (security and computing) може дати істотний практичний результат для підвищення якості сервісів, життя і збереження екології планети. Згадані джерела побічно зачіпають питання активного кіберфізичного комп'ютинга, пов'язаного з актюаторним управлінням кібербезпеки. Позбавлення людини функції управління безпекою та передача її кіберфізичному CSC-комп'ютинга є найголовнішою організаційною проблемою креативного світу. Тому громадянин, соціальна група, компанія, держава і людство потребують створення масштабованого аватара у форматі Gartner-computing: «security assistent – digital twin – smart security robot», який позбавить людей невірних рішень, що призводять до небажаних наслідків на ринку кіберфізичних і соціальних технологій.

Комп'ютинг – галузь знань, яка займається розвитком теорії і практики надійного метричного управління віртуальними, фізичними (природними) і соціальними процесами і явищами на основі використання комп'ютерних дата-центрів і мереж, великих даних і цифрового моніторингу кіберфізичного простору за допомогою інтелектуальних пошуково-аналітичних сервісів, персональних гаджетів і розумних датчиків.

Комп'ютинг (рис. 1) – процес моніторингу (5) і актюації (6) метричних відношень (2) в інфраструктурі управління (3) і виконання (4) для досягнення і візуалізації (8) мети – продукції (1) при заданих ресурсах (7).

Рис. 1. Комп'ютинг

Cyber Security Computing (CSC) – галузь знань, що займається розвитком теорії і практики надійного метричного online управління кібер-захисту великих даних, віртуальних, фізичних (природних) і соціальних процесів і явищ в комп'ютерних дата-центрах і мережах на основі точного цифрового моніторингу деструктивних компонентів кіберфізичного простору за допомогою інтелектуальних пошуково-аналітичних сервісів і розумних сенсорів.

Метричне визначення комп'ютинга за допомогою восьми взаємопов'язаних компонентів надає теоретичну фундаментальну основу для формального і фактичного створення будь-якого процесу в заданій сфері людської або природної діяльності. Види комп'ютинга за введеною метрикою охоплюють всі сфери люд-

ської діяльності: космологічний, біологічний, флористичний, фізичний, віртуальний, кібербезпечний, квантовий, соціальний, державний, медичний, транспортний, інфраструктурний, науковий, освітній, виробничий, спортивний, відпочинку, подорожей, розваг.

Процес – матеріально-енергетична взаємодія системних компонентів у часі і просторі для досягнення мети. Глобально: процес – матеріально-енергетична зміна в просторово-часовому континуумі. Локально: процес є розвиток просторового відношення компонентів (явищ) у часі.

Явище – компонент (системи) або фрагмент процесу в фіксований момент часу, що сприймається рецепторами, почуттями, вірою або розумом.

Cyber Security Computing: процес – спостережувана взаємодія механізмів управління та виконання в часі і просторі на основі моніторингу та актюації метричних відносин між програмними додатками і деструктивним кодом для досягнення мети у вигляді тестування, діагностування та усунення різного виду malware.

Відношення – структура взаємопов'язаних компонентів, що визначає властивості процесу або явища. Структура визначає властивості компонентів, але ніяк не навпаки. Первинно відношення-сигнатура, вдруге носії-компоненти. Алфавіт є носієм відношення, що визначається за допомогою операцій (сигнатури) над символами. Просто символи алфавіту не мають сенсу. Відношення є визначальними при створенні ефективних: математичних теорій, структур даних, алгоритмів, архітектур, моделей, методів, технологій, програмно-апаратних додатків, кіберфізичних і соціальних систем, включаючи економіку, охорону здоров'я, транспорт, юриспруденцію, охорону правопорядку, кібербезпеку, екологію і державність. Потужність відношення, як інтегральна сукупність і якість взаємних зв'язків між компонентами, формує метрику, яка дає можливість ідентифікувати ефективність структури.

Метрика відношень. Елементарна основа світобудови є відношення між двома компонентами: процесами або явищами. Як правило, в процесі - це відношення нерівності пари компонентів, яке вимірюється ставленням рівності (xor, not-xor), що становить сутність метрики.

Не існує компонента без відношення, оскільки дана процедура є відношенням між двома компонентами. Це вірно і для випадку, коли сам компонент знаходиться у відношенні рефлексивності з собою. Тому елемент визначається і розглядається як частина відношення.

Метрика первинності відносини нерівних компонентів поширюється глобально на всі явища і процеси, пояснює їх і породжує їх: 1) Одиниця – Нуль. 2) Чорне – Біле. 3) Багатий – Бідний. 4) Добро – Зло. 5) Розумний – Дурень. 6) Студент – Учитель. 7) Керівник – Виконавець. 8) Елемент – Множина. 9) Чоловік – Жінка. 10) Моніторинг – Управління. 11) Курка – Яйце. 12) Простір – Час. 13) Матерія – Енергія. 14) Reading – Writing. 15) Listening – Speaking. 16) Процес – Явище. 17) Хаос – Порядок. 18) Еліта – народ. 19) Живе – Неживе. 20) Software – Malware.

Взаємодія протилежних асиметричних явищ у часі створює стійку структуру або процес еволюції. Взаємодія синонімічних явищ в часі створює нестійку структуру або процес деградації. Похідна по антонімічним або протилежним явищ чи процесів дорівнює їх об'єднанню. Похідна за всіма 20 згаданим парам відношень дорівнює їх об'єднанню. Симетрія або еквівалентність компонентів відношення не здатна до еволюції. Тому відношення рівності компонентів у системі означає кінець розвитку. Критерієм даного факту є нульове значення похідної між взаємодіючими компонентами системи. Компоненти відношень в процесі еволюції системи перетворюються один в одного. Відношення породжує елементи, але не навпаки. Мета процесу – еволюційний перехід з одного явища в інше.

CSC – процес тестування, моніторингу, діагностування та активації сигналів деструкції шкідливих компонентів на основі метричних відносин між malware і software в кіберфізичному просторі.

CSC-процес – спостережувана взаємодія механізмів malware і software в часі і просторі на основі моніторингу та актюації метричних відносин для досягнення мети у вигляді усунення malware при виділених ресурсах.

Malware-функціональність (MF) являє собою структуру взаємопов'язаних логічних елементів, яка забезпечує цифрову реалізацію деструктивної поведінки об'єкта в заданому просторі software змінних.

Malware-змінна (MV) визначається упорядкованим універсумом примітивних значень, який формує проекцію поведінки об'єкта на векторі змінних, що створює malware-функціональність.

Логічний malware-елемент (ML) являє собою еталонне відображення значень багатозначної змінної в двійковий кубітний вектор, заданий на упорядкованому універсумі примітивних значень.

Значення (LV) змінної – унікальна примітивна властивість об'єкта, що має пустий перетин з іншими примітивами, яке в суперпозиції з ними складає універсум.

Таким чином, проглядається структурована ієрархія введених понять (рис. 2):

(CSC – MF – MV – ML – LV),

яка формує можливі архітектурні рішення malware-комп'ютингу.

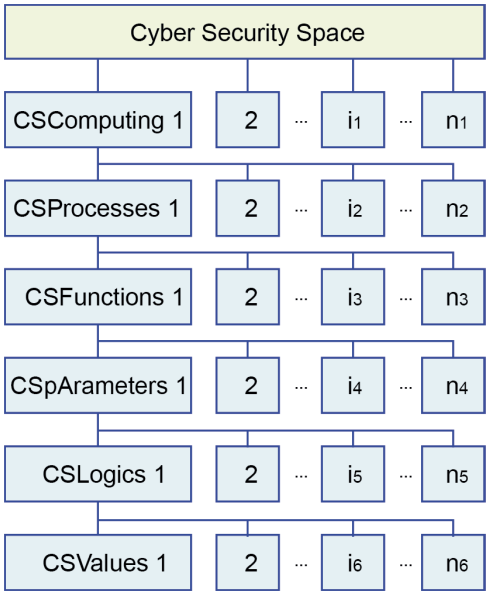

Рис. 2. Ієрархія malware-комп'ютинга

ML-рівень архітектури характеризується синтезом логічної схеми, де кожен елемент має одну багатозначну регістрову змінну, яка фактично представлена кубітним вектором, де число одиничних координат може бути більше одиниці. Ця властивість кубіта дає можливість створювати компактні структури даних malware-функціональностей з метою їх паралельної обробки. Для виконання квантового методу моделювання на кубітних структурах даних необхідно унітарно закодувати вхідні символьні дані за допомогою таблиць-універсумів значень, що відповідають кожній змінній.

Змінна ототожнюється з ключовим словом-поняттям (keywords), яке найбільш часто зустрічається в потоці вхідних даних. Набір таких keywords створює непересічну множину змінних в malware-процесі, де їх значення представлені синонімами ключових слів, які формують багатозначність змінної як клас еквівалентності.

Сукупність змінних створює простір malware-процесу, в якому визначаються еталонні, практично орієнтовані функціональності malware-комп'ютинга у вигляді логічних кубітних схем для моделювання вхідних потоків даних, взятих із хмари, мережі, комп'ютерів або гаджетів.

Мета – створення логічного кубітного процесора для паралельного моделювання та розпізнавання malware-функціональностей у вхідних потоках великих даних, отриманих шляхом метричного моніторингу інфраструктурних компонентів, для подальшого цифрового управління актюаторними сигналами по деструкції шкідливих компонентів.

Завдання пов'язані зі створенням моделі, методу і процесора malware-комп'ютингу, спрямованого на автоматичний синтез і аналіз кубітних логічних схем, орієнтованих на моніторинг, моделювання, розпізнавання і деструкцію шкідливих кодів.

1) Процесор malware-комп'ютингу на основі синтезу кубітних логічних схем для моніторингу, моделювання, розпізнавання і деструкції шкідливих кодів.

2) Кубітно-векторна модель опису універсуму значень багатозначної змінної для синтезу логічного секвенсора, орієнтованого на аналіз malware-компонентів.

3) Кубітний метод синтезу логічної схеми для моделювання та розпізнавання malware-функціональностей на основі унітарного кодування значень багатозначної змінної.

4) Кубітний метод аналізу malware-компонентів на основі використання еталонних логічних елементів malware-функціональностей з унітарним кодуванням багатозначних змінних.

5) Тестування і верифікація кубітних моделей і методів malware-комп'ютинга на прикладах аналізу і розпізнавання malware-процесів у вхідних потоках великих даних, пов'язаних з кіберфізичним відображенням обчислювальних процесів.

2. Квантові процеси і явища

Gartner тенденції світової кіберкультури [3-14] (рис. 3) формують технологічну культуру для створення глобального кіберфізичного CSC-комп'ютинга в рамках укладу Internet of Things.

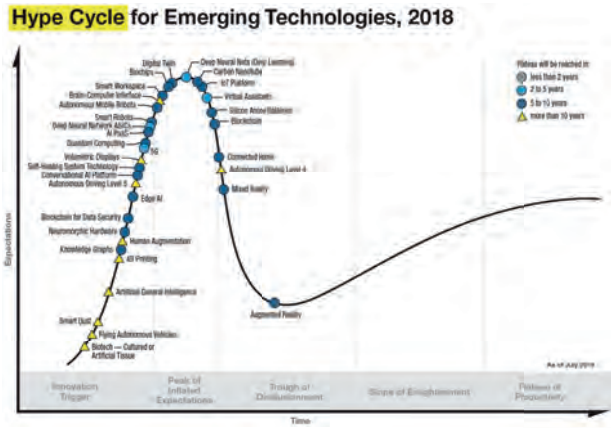

Рис. 3. Gartner топ-технології 2018

При цьому квантовий комп'ютинг розглядається як енергозберігаюче майбутнє цифрового світу, створюваного для підвищення якості життя і збереження екології планети. Зокрема, квантовий паралельний комп'ютинг і кубітні структури даних дозволяють спростити алгоритми в області cyber security computing і підвищити швидкодію програмних продуктів на класичних комп'ютерах.

Стратегічні тенденції в області цифрових технологій [3-7] приведуть у 2020 році до суттєвих дізрапцій, що надасть нові можливості розробникам корпоративної архітектури і конструктивних інновацій з метою створення конкурентних переваг при використанні нових трендів кіберкультури: 1) Автономні фізичні та віртуальні інтелектуальні та координовані речі. 2) Розширена, доповнена кіберпростором, соціальна CSC-аналітика для вироблення актюаторних впливів. 3) AI-кероване проектування, розширений і автоматичний розробник. 4) Цифрові близнюки; цифровий образ організації або компанії. 5) Взаємодоповнюючі один одного Edge + Cloud Computing. 6) Досвід занурення в цифрову дійсність і своєчасне сприйняття деструктивних змін в цифровому світі. 7) Використання Blockchain в соціальній схемі. 8) Smart Spaces. 9) Цифрова етика і конфіденційність особистого життя. 10) Квантові обчислення, квантова кібербезпека, розвиток і становлення квантового комп'ютинга.

Визначення «квантовий» характерно не тільки для субатомних структур, взаємодіючих за моделлю (комплексних) гільбертового простору квантової механіки. Дане визначення у теперішній час носить глобальний характер і імплементується в усі сфери науки, освіти і людської діяльності.

Вчені і практики планети створюють сьогодні новий квантовий технологічний уклад, який формується на основі класичних наук, що представлені квантовою електродинамікою, механікою, фізикою та математикою, і на якому успішно приростають і розвиваються нові дисципліни і галузі знань: квантовий комп'ютинг, квантові структури даних, квантові схеми, квантові алгоритми, квантова логіка, квантове моделювання схем, квантові групи, квантове моделювання (quantum computer modeling – IEEE Xplore 3436), квантові розподілені системи, квантова інформация, квантові телекомунікації, квантові системи, квантові точки, квантова метрологія, квантова радіотехніка, квантові комп'ютерні науки, квантова криптографія і кібербезпека, квантова телепортація, квантова компресія даних, квантова пам'ять, квантова фотоніка,

Quantum computer modeling & quantum-driven algorithm everywhere, як інженерні рішення, використовує технологічну ідею квантового змішування і суперпозиції для синтезу продуктивних паралельних алгоритмів memory-driven комп'ютингу в усіх сферах людської діяльності.

Ефективність квантових обчислень на класичному комп'ютері визначається метрикою паралельного рішення комбінаторних задач на кубітних структурах даних за рахунок ускладнення алгоритмів і збільшення пам'яті для зберігання унітарних кодів.

Квантовий комп'ютинг оперує двома базовими технологічними операціями: суперпозиція і переплутування, яким ставляться у відповідність логічні операції функціонально повного базису: диз'юнкція і інверсія в класичному обчислювачі.

Основа квантової технології визначається квантовою невизначеністю електронів, суперпозиція яких дає можливість отримати в одній точці простору будь-який універсум кінцевого числа примітивних сигналів, включаючи нуль і одиницю. Ізоморфним аналогом виступає дискретна невизначеність Кантора X={0,1}, яка використовується для отримання компактних форм структур даних, а також для паралельного виконання теоретико-множинних операцій при унітарному кодуванні елементів універсуму.

Приклад quantum-driven алгоритму: знайти в множині елементів Q число X або визначити належність X∈Q. Рішення зводиться до перебору порівнянь числа X з усіма числами масиву Q, кількість яких дорівнює n. Обчислювальна складність такої процедури дорівнює C=n. Квантове рішення даної задачі зводиться до унітарного кодування всіх чисел з метою синтезу структури даних у формі одного вектора в форматі універсуму-множини чисел, де кожно-

му з них відповідає середнє арифметичне значення координати, адреса якої дорівнює числу, присутньому в множині. Після цього виконується операція: if Q(X)=1 then Y=1, або простіше Y=Q(X), де і є число, яке підлягає визначенню належності X∈Q. Значення Y=1 є індикатором належності числа до заданої множини: X∈Q. Обчислювальна складність процедури пошуку числа в множині пов'язана з побудовою кубітного вектора, на що витрачається n операцій, і з виконанням логічної перевірки вмісту комірки за адресою-числом, що надійшов на вхід квантового секвенсора: C=n+1. Дійсно, разове використання такого обчислювача не дає збільшення швидкодії. Однак багато разів (m) повторена процедура пошуку числа в неврегульованому масиві при великих значеннях m призводить до одиниці обчислювальну складність алгоритму, включаючи первинне ініціювання вектора унітарних кодів чисел. Даний факт дає підстави вважати обчислювальну складність процедури по багаторазовому пошуку числа в масиві, рівну одиниці: C=(n+m)/m.

Приклад визначення належності деякої підмножини елементів X множини Q. Рішення: після унітарного кодування елементів обох множин виходять два вектора однакової довжини, до них застосовуються лише три векторно-регістрових паралельних операції (not, and, or):

$$Y = \vee[(X \wedge Q) \oplus X] = \vee[(X \wedge \neg Q)],$$

які разом з препроцесінгом унітарного кодування мають обчислювальну складність C=2n+3. Класичний варіант вирішення даної задачі має обчислювальну складність, яка дорівнює

$$C = (1/2) * (n-1) ** 2.$$

Вже при n = 7 квантовий або кубітний метод починає вигравати в продуктивності рішення даного завдання перед класичним.

3. Кубітні моделі кіберфізичного соціального CSC-комп'ютинга

Пропонуються архітектури і класичні структури, пов'язані з кіберфізичним комп'ютингом (метричний моніторинг і цифрове управління), спрямованим на прийняття рішень, пошук і ідентифікацію malware, визначення функцій належності вхідних даних до заданого деструктивного зразка на основі введеної метрики визначення відстаней. Всі моделі орієнтовані на схемотехнічну реалізацію методів і алгоритмів online моделювання з метою вироблення human-free адекватних автоматичних актюаторних впливів, спрямованих на знищення деструктивних кодів. Логічні кубітні структури здатні розпізнавати деструктивні коди, що надходять на вхід спеціалізованого комп'ютера, і усувати їх з програмних додатків.

Кожен malware-параметр може мати свою альтернативу значень в позитиві та негативі (мультиверсність також допускається), тоді їх число подвоюється. Але можна використовувати тільки позитивні зразки, засновані на конструктивних параметрах або атрибутах. Такі образи – логічні процесори – формують еталонні якості malware-процесів і явищ.

Квантові технології паралельних обчислень ефективно використовуються для вирішення комбінаторних проблем, емулюючи обчислення на класичних комп'ютерах [14-16]. З іншого боку, таблиці істинності або кубічні покриття для опису логічних елементів є ефективними структурами даних для вирішення проблем CSC-комп'ютинга та пошуку необхідних даних [17, 18]. Автоматичний синтез кубітних покриттів функціональностей є одним з основних важко формалізованих завдань, без якого неможливо виконувати аналітику великих даних [19-22]. Для цих цілей далі вводиться аналітична модель W кубітно-логічного процесора CSC-комп'ютинга, яка оперує двома матрицями: універсумів U примітивів і кубітних функціональностей Q, а також логічними примітивами L, що інтегрують функціональності в комбінаційну схему CSC-процесора:

$$W = (U, Q, L),$$
$$U = (U_1, U_2, ..., U_i, ..., U_n);$$
$$\bigcup_{i-1}^{n} U_i = U;\ U_i \bigcap_{i,k=1,n}^{i \neq k} U_k = \varnothing;$$
$$Q = (Q_1, Q_2, ..., Q_i, ..., Q_n);$$
$$\bigcup_{i-1}^{n} Q_i = Q;\ Q_i \bigcap_{i,k=1,n}^{i \neq k} Q_k = \varnothing;$$
$$Q_i = (Q_{i1}, Q_{i2}, ..., Q_{ij}, ..., Q_{im});\ Q = [Q_{ij}];$$
$$U_i = (U_{i1}, U_{i2}, ..., U_{ij}, ..., U_{im});\ U = [U_{ij}];$$
$$L = f[Q] = (Q_1 \circ Q_2 \circ, ..., \circ Q_i \circ, ..., \circ Q_n)$$
$$\circ = \{\wedge, \vee, \oplus\};$$
$$U_{ij} \in U_i \in U;\ Q_{ij} \in Q_i \in Q;\ Q_i \in U_i;\ Q \in U;$$
$$Q_{ij} = 1 \leftarrow \max \mu(R, U_{ij}).$$

Метрика-універсум U тут виконує роль еталонного зразка для порівняльного аналізу еталона з вхідним потоком деструктивних кодів і даних R=X, що реалізується за допомогою аналізатора-компаратора, що видає максимальне значення функції належності, яке трансформується в

одиницю на відповідній координаті одного з кубітів

$$Q_{ij} = 1 \leftarrow \max \mu(R, U_{ij}).$$

Архітектура метричної взаємодії U-матриці універсумів з потоком даних R для обчислення функцій належності μ(R, U), з метою отримання Q-матриці значень і подальшого L-об'єднання кубітів в комбінаційну схему кіберсоціального процесора, представлена на рис. 4.

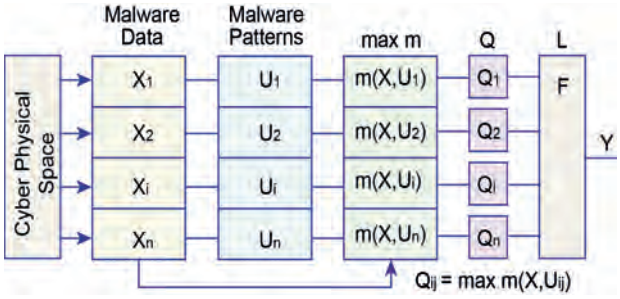

Рис. 4. Архітектура для синтезу CSC-процесора

Тут вхідний потік модельованих великих деструктивних даних R має такий же формат, як U-, Q-матриці і комбінаційна схема процесора. Алгоритм синтезу Q-матриці полягає у визначенні максимального значення функції належності вхідного фрейму розглянутої змінної до одного з значень відповідного рядка матриці універсумів. В результаті такого порівняння по всіх координатах U-матриці формуються поодинокі координати кубітної матриці, де кожен рядок являє собою примітивну функціональність по розглянутій змінної. Разом всі рядки Q-матриці створюють комбінаційну схему логічного CSC-процесора для моделювання будь-якого вхідного впливу з метою визначення його належності до даного еталону деструктивного процесу або явища.

Процедура синтезу кубітів необхідна для аналізу malware-потоків даних шляхом використання логічних CSC-еталонів. Фрагменти даних надходять на входи одного або декількох логічних елементів, що формують метрику CSC-процесу:

$$U_{ij} \in U_i \in U;\ Q_{ij} \in Q_i \in Q;\ Q_i \in U_i;\ Q \in U;$$

В результаті моделювання вхідного потоку деструктивних даних формуються бінарні значення переваг еталона в кубітному векторі кожного логічного елемента, відповідного одному параметру. Для цього використовується метричний вимір функції належності вербальних даних до кожного значення наперед заданого універсуму примітивів логічної змінної. Так автоматично створюються кубітні зразки CSC-функціональності.

Формування повної множини параметрів-змінних CSC-процесу або явища також пов'язано з аналітикою великих даних, спрямованою на отримання ключових понять-слів, максимально віддалених один від одного по метриці (кодової відстані) класів еквівалентності, які покривають всі CSC-змінні. Слід зауважити, що універсум примітивів ототожнюється з класом непересічних еквівалентностей, які створюють всі можливі значення даної змінної-класу в той час, як множина еквівалентних класів відповідає універсуму змінних вищого рівня ієрархії. Дані властивості використовуються при аналітичному синтезі універсуму змінних, що покривають цифровий образ CSC-процесу гранями, які формують еталон-функціональності процесу або явища.

Наприклад, необхідно синтезувати віртуального CSC-асистента (virtual assistent) або цифрового двійника (digital twin), або розумного робота (smart robot), який буде реагувати на зовнішні вхідні дані. Алгоритм для створення аватара містить такі кроки: 1) Синтез універсуму змінних-примітивів, що покривають всі функціональності CSC-процесу. 2) Синтез U-матриці універсумів значень-примітивів, що покривають всі можливі варіанти кожної змінної, в рамках CSC-процесу або явища. 3) Синтез Q-матриці конкретних значень-примітивів у форматі кубіт-вектора кожної змінної в рамках CSC-процесу або явища. 4) Перевірка отриманої U-матриці універсумів CSC-процесу або явища на повноту і примітивізм змінних і значень.

Після синтезу кубітних векторів за всіма параметрами в Q-матриці всі значення виходів кубітних елементів надходять на входи інтегратора L, що працює по функції and (може бути і інша функція, наприклад, not-and), який видає два значення {1,0}: позитивний або негативний результат моделювання, який можна інтерпретувати як бінарну функцію належності до ідеалу malware-функціональності, формує, наприклад, властивості CSC-процесу.

Таким чином, logic-процесор, синтезований на основі використання Q-матриць квантових структур даних, здатний online моделювати будь-які CSC-процеси і явища, недоступні сьогодні для класичного комп'ютинга в базисі традиційних логічних елементів, зважаючи на складність формалізації поведінки malware для синтезу цифрових моделей-схем.

Формалізм створення еталон-схеми для CSC-процесу або явища полягає у визначенні числа істотних параметрів, де всередині кожного з них генерується сукупність значень, унітарно кодованих для синтезу кубітного вектора логічного елемента. Логічні примітиви, що відповідають істотним параметрам, об'єднуються за функціями (and, or, not, xor), які регулюють взаємні відносини між параметрами для формування кінцевого результату про валідність вхідного процесу або явища по відношенню до одного або декількох стандартів.

CSC-комп'ютинг може бути представлений як кіберфізична система інтелектуального хмарного управління CSC-процесами на основі точного цифрового моніторингу: розумної електронної інфраструктури; співробітників компанії, оснащених комп'ютерами та персональними гаджетами; транзакцій і процесів, заданих в часі і просторі. Структура системи CSC-комп'ютинга представлена трьома взаємодіючими макрокомпонентами: хмарне інтелектуальне управління, електронна CSC-архітектура, кіберфізичний простір (рис. 5).

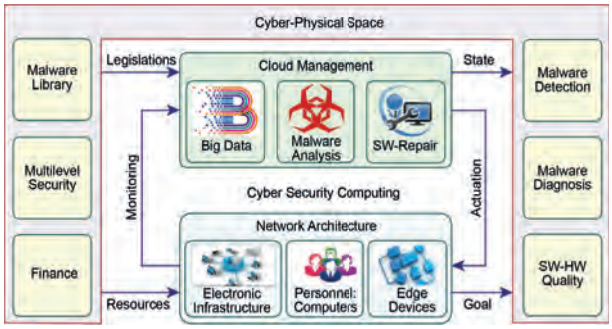

Рис. 5. CSC-комп'ютинг моніторингу та управління процесами

Хмарні компоненти-сервіси моніторингу та усунення malware працюють за схемою: факт – оцінка – дія. Тут виконується знімання великих даних з різних розумних сенсорів і комп'ютерів, інтелектуальний аналіз даних на основі CNN, DNN, ML. Останнім компонентом хмарного сервісу є формування цифрових актюаторних впливів, орієнтованих на управління інфраструктурними компонентами, для досягнення мети (Goal) у вигляді видалення malware і забезпечення високої якості кіберпослуг. Вся система CSC-комп'ютинга безпосередньо взаємодіє з кіберпростором або інтернетом, який обов'язково є входом і виходом для створюваної структури. Крім того, входами є Legislations, які формують відносини в компанії, а також Resourses у вигляді фінансів і матеріалів, необхідних для захисту процесів створення продукції і/або сервісів. Важливим виходом системи є State, який ідентифікує стан розвитку комп'ютерної системи.

CSC-комп'ютинг є технологією ефективного хмарного управління для істотного зниження накладних непродуктивних витрат і підвищення прибутку, яка характеризується оперативним online моніторингом процесів на основі використання сучасної кіберкультури, що включає: IoT, Cyber Physical Systems, Cloud Computing, e-Infrastructure, Big Data Analytics , Artificial Intelligence, Blockchain smart contract, e-Dicument Circulation and Internet.

Принципи реалізації: 1) Моніторинг програмних додатків за допомогою впровадженого агента в умовах інваріантності геопозиції. 2) Моніторинг всіх пристроїв для інтелектуального аналізу і подальшого управління структурними компонентами, дає можливість оперативно приймати рішення по реконфігурації CSC-процесів у реальному часі. 3) Моніторинг, замкнутий на online управління, без активної участі людини. У цьому сьогодні головне і ще не вирішене завдання CSC-ринку.

Моніторинг malware без актюаторних впливів на усунення деструктивних компонентів, що виробляються кіберфізичною human-free CSC-системою, не представляє ринкового інтересу з позиції сучасної кіберкультури. Рішення проблеми цілком очевидне – створення CSC-системи моніторинга, але головне – online управління CSC-процесами на основі створення розумних алгоритмів або смарт-контрактів, що програмують легітимні відносини в кіберпросторі. Програмний код реалізує тріаду соціальних подій, без участі чиновника: факт – оцінка – дія, яка модельно зводиться до кодування алгоритму обробки вхідних даних для отримання вихідних актюаторних впливів, спрямованих на компоненти CSC-інфраструктури, яка виконується в рамках технологічного укладу IoT. Компонентами CSC-системи є: 1) Відносини, прийняті в компанії (державі) на основі існуючого законодавства, статуту (конституції), наказів, традицій, історії, культури. 2) Мета та/або напрямок руху, зрозумілі для ринку, що мобілізують співробітників для якісного виконання завдань. 3) Цифровий менеджмент або управління – секретний ключ ринкового успіху, – обов'язково використовує хмарні сервіси, максимально виключає участь людини в моніторингу виробничих процесів і подальшо-

му прийнятті рішень. 4) Інфраструктура підприємства, що забезпечує комфортні умови для конструктивної роботи, якісного харчування та активного відпочинку в форматі 24/7: onsite & remote online. 5) Кадри, що створюють ринкову продукцію та послуги, – головне надбання або інтелект будь-якої компанії, який оцінюється симетричною різницею компетенцій співробітників [17]:

$$I = \bigoplus_{i=1}^{n} P_i = \bigcup_{i=1}^{n} P_i \leftarrow P_i \cap P_j = \varnothing;$$

$$I = \bigwedge_{i=1}^{n} P_i = \bigcap_{i=1}^{n} P_i \leftarrow P_i = P_j. \quad \forall i, j$$

4. Унітарно-кодовані структури даних. Сигнатурний аналіз malware

Інтерес представляє проблема аналізу великих даних з метою встановлення нових malware-функціональностей на природному тлі вже визначених. Аналогічна задача була вирішена в Лабораторії Касперського (ЛК) і вирішується досі засобами роботів і експертів в області malware and virus аналітики. Тут використовується сигнатурний аналіз, адаптований до вірусної аналітики, який дозволяє мати досить компактний код-сигнатуру деструктивності з метою високопродуктивної ідентифікації в потоці великих даних старих вірусів. Це дає можливість акцентувати увагу робота-експерта на детальному аналізі нових деструктивів з метою їх подальшого блокування. Перекладаючи згадану сигнатурну технологію на рішення проблем malware-аналітики, слід зазначити важливість отримання компактної таблиці malware-функціональності, інваріантної до часу. Першим кроком в цьому напрямку є мінімізація таблиці унітарного кодування malware-функціональності за дозволеними логічними правилами (суперпозиціями), які дають можливість отримати один стовпець, що ідентифікує malware-функціональність. Структурні протиріччя при об'єднанні координат стовпців унітарно-кодованої матриці відсутні. Природно, що в результуючому стовпці-сигнатурі буде втрачена структурна інформація про порядок виконання сервісу, що є платою за компактність і швидкодію по ідентифікації стовпців malware-функціональності. Однак структурна інформація не стирається і може бути затребувана в разі необхідності. Теоретичним підтвердженням і обгрунтуванням запропонованої суперпозиційної інновації зі стиснення стовпців в один є той факт, що кодування будь-якої таблиці істинності двома і навіть одним кубітним вектором, отриманим за допомогою суперпозиції унітарних кодів вхідних впливів будь-якого, як завгодно складного цифрового пристрою (рис. 6). Обмеження: всі атрибути в матриці унітарного кодування, що підлягають суперпозиції по конкретних даних, повинні бути незалежними один від одного. Суперпозиція стовпців унітарної матриці дає можливість отримати покриття всіх атрибутів-змінних одиничними значеннями різноманітності даних. Якщо одиницями покриті в повному обсязі значення атрибутів, то існує некоректність в аналізі та кодуванні даних по конкретному атрибуту. В масштабах метрики значень інтегральний стовпчик **malware**-функціональності завжди буде являти собою підмножину з нульових та одиничних координат на фоні повністю одиничних значень інтегрального стовпця універсуму P∈R if P∩R=P.

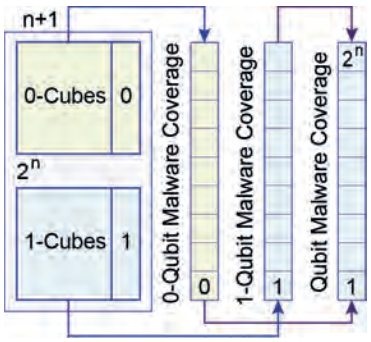

Рис. 6. Унітарне кодування універсуму примітивів по кожній змінній

Суперпозіційна модель уявлення malware-функціональності інваріантна до часу. Ідея класифікації полягає в порівнянні великих даних з інтегральним вектором, який виходить шляхом суперпозиції або об'єднання всіх стовпців malware-функціональності

$$P = \bigcup_{i=1}^{n} P_i.$$

Процедура ідентифікації зводиться до операції перетину між стовпцем вхідних даних і інтегральним стовпцем malware-функціональності: $S \in P \leftrightarrow S \cap P = S,$ яка повинна дорівнювати вектору вхідних даних. Природно, виникнуть ситуації, коли не буде виконуватися наведена вище умова за всіма порівняннями за допомогою стовпців malware-функціональності. Тоді слід керуватися правилом домінування мінімальної кодової відстані за Хемінгом:

$$S_i \in P_j \leftrightarrow \min(S_i \overset{m}{\underset{j=1}{\cap}} P_j = \varnothing), i = \overline{1,n};$$

$$S_i \in P_j \leftrightarrow \min(S_i \overset{m}{\underset{j=1}{\wedge}} P_j = 1), i = \overline{1,n}.$$

Для аналізу детермінованої двійкової моделі malware-функціональності існує ефективний апарат булевих похідних, який визначає істотність і неістотність змінних щодо формування вихідного значення функціональності. Якщо зміна стану змінної-атрибута не призводить до зміни функціональності, то така змінна є несуттєвою і її можна виключити з моделі CSC-функціональності.

При вербальному завданні моделі malware-функціональності розробники використовують свій досвід і інтуїцію для формування екстра-функціональностей, дублюючих деякі істотні атрибути на моделі malware-функціональності. Дана, в межі 100%-ва, надмірність може бути використана також для асерціонної верифікації моделі CSC-процесу. Сенс такої верифікації полягає в незалежному створенні і подальшому використанні-порівнянні двох моделей, де перша – максимально точна за всіма параметрами, друга – створює картину станів істотних змінних. Асерції, що не несуть нової інформації про модель, дають можливість уточнити наявність стану даних для істотних атрибутів без malware-функціональності в конкретному часовому фреймі.

Модельна надмірність, як правило, є корисною для прискорення обчислювальних процесів за рахунок диверсифікації структур даних. Наприклад, просторово-часова модель malware-функціональності за рахунок конволюції часу в точку може бути компактно представлена одним інтегральним стовпцем даних.

Багатозначність параметрів malware-функціональності укладається в таку матричну модель:

$$P = [P_{ij}], i = \overline{1,n}; j = \overline{1,m};$$

$$P = (P_1, P_2, ..., P_i, ..., P_n);$$

$$P_i = (P_{i1}, P_{i2}, ..., P_{ij}, ..., P_{im}).$$

Тут n – число рядків матриці malware-функціональності; m – кількість значень параметра Pi при її кодуванні.

Для оптимізації malware-функціональності необхідно і достатньо використовувати відомі аксіоми алгебри логіки:

1) $a \vee a = a \leftrightarrow 1 \vee 1 = 1;$

2) $a \vee ab = a(1 \vee b) = a \leftrightarrow 1x \vee 11 = 1;$

3) $ab \vee a\bar{b} = a(b \vee \bar{b}) = a \leftrightarrow 11 \vee 10 = 1x = 1;$

4) $abc \vee b = b.$

Логічні аксіоми трансформуються в закони теорії множин, де фігурують елементи в форматі унітарних кодів значень вхідних змінних:

1) $a \cup a = a;\ a \cap a = a;$

2) $a \cup ab = a(1 \cup b) = a \leftrightarrow 1x \cup 11 = 1x = 1;$

3) $ab \cup a\bar{b} = a(b \cup \tilde{b}) = a \leftrightarrow 11 \cup 10 = 1x = 1;$

4) $abc \cap b = b \leftrightarrow 111 \cap 010 = 010.$

Всі вербальні значення або частини істотних (додаткових) параметрів повинні бути унітарно і єдино-метрично закодовані з метою представлення координат матриці malware-функціональності двійковими векторами, які дають можливість в паралельному режимі визначати належність вхідного вектора одному або кільком стовпцям malware-функціональності шляхом застосування логічної процедури :

$$a \in ab \leftrightarrow a \cap ab = a \rightarrow 10 \cap 11 = 10 \wedge 11 = 10.$$

У загальному випадку метрична взаємодія двох компонентів однієї розмірності може мати тільки п'ять випадків (рис. 7) [19]:

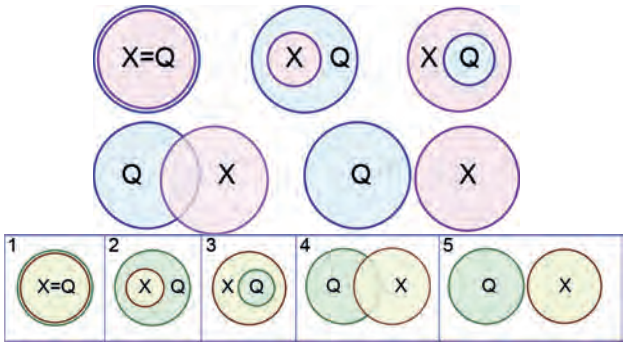

Рис. 7. Взаємодія даних по and-операції

1) Належність чи рівність об'єктів один одному, якщо виконується умова:

$$a = b \leftrightarrow a \cap b = \{a, b\} \rightarrow 10 \cap 10 = 10 \wedge 10 = 10.$$

2) Належність першого A другому m, якщо виконується умова:

$$a \in b \leftrightarrow a \cap b = a \rightarrow 10 \cap 11 = 10 \wedge 11 = 10.$$

3) Належність другого m першому A, якщо виконується умова:

$$b \in a \leftrightarrow a \cap b = b \rightarrow 11 \cap 10 = 11 \wedge 10 = 10.$$

4) Часткова належність об'єктів один одному, якщо виконується умова:

$$a \neq b \leftrightarrow a \cap b \neq \{\varnothing, a, b\} \rightarrow 110 \cap 011 = 110 \wedge 011 = 010.$$

5) Неналежність об'єктів один одному, якщо виконується умова:

$$a \neq b \leftrightarrow a \cap b = \varnothing \rightarrow 01 \cap 10 = 01 \wedge 10 = 00.$$

Структурна карта модулів комп'ютинга для аналізу CSC-процесів (рис. 8):

1) Синтез матриці істотних змінних.

3) Побудова унітарної матриці даних.

5) Декомпозиція унітарної матриці даних.

6) Синтез M-RPA (Malware Robotic Process Automation) на основі застосування ML-технології до матриць malware-функціональностей.

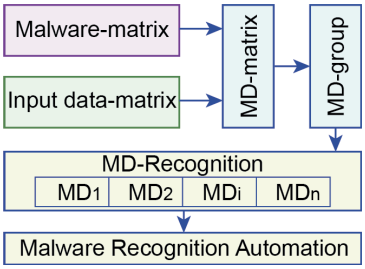

Рис. 9. Структура для аналізу malware-процесів

Визначення унітарної матриці істотних змінних malware-процесів і кодування всіх значень.

Розв'язок. Організовується цикл по n змінним без malware-функціональності, де всередині створюється цикл за значеннями змінних, де є ще один вкладений цикл, що перелічує всі існуючі malware-функціональності, які обробляються на предмет їх оригінальності (рис. 9). Таким чином, програмний модуль P-matrix, що містить три вкладених цикли, створює таблицю відповідності текстових значень істотних змінних їх десятковим номерам або унітарним кодам для подальшого аналізу CSC-процесів.

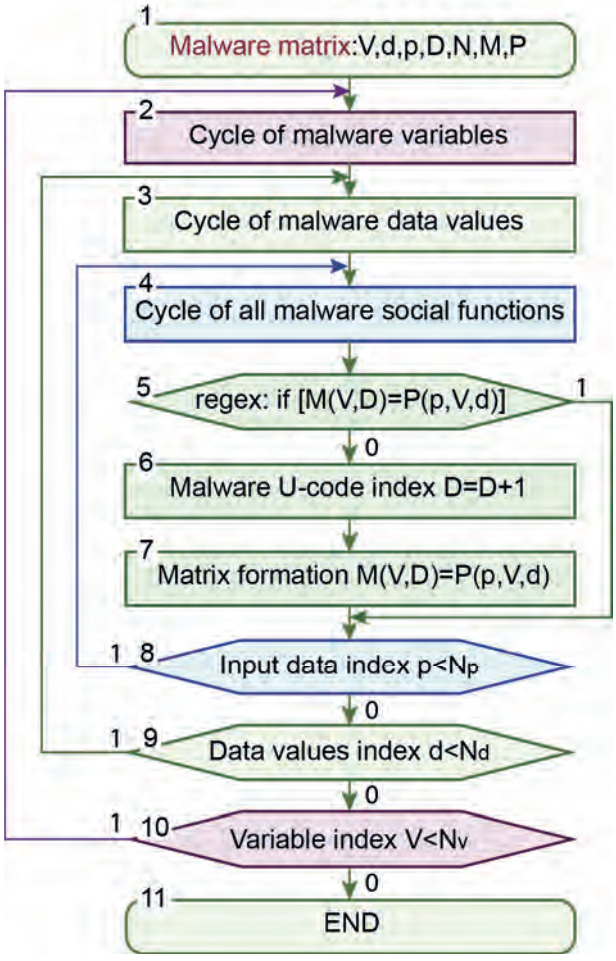

Рис. 9. Алгоритм формування матриці значень параметрів

У загальному випадку метрична взаємодія двох стовпців-malware однієї розмірності може мати тільки п'ять випадків:

1) Належність чи рівність malware один одному, якщо виконується умова:

$$a = b \leftrightarrow a \cap b = \{a, b\} \rightarrow 10 \cap 10 = 10 \wedge 10 = 10.$$

2) Належність першого malware A другого m, якщо виконується умова:

$$a \in b \leftrightarrow a \cap b = a \rightarrow 10 \cap 11 = 10 \wedge 11 = 10.$$

3) Належність другого malware m першому A, якщо виконується умова:

$$b \in a \leftrightarrow a \cap b = b \rightarrow 11 \cap 10 = 11 \wedge 10 = 10.$$

4) Часткова належність malware один одному, якщо виконується умова:

$$a \neq b \leftrightarrow a \cap b \neq \{\varnothing, a, b\} \rightarrow 110 \cap 011 = 110 \wedge 011 = 010.$$

5) Неналежність malware один одному, якщо виконується умова:

$$a \neq b \leftrightarrow a \cap b = \varnothing \rightarrow 01 \cap 10 = 01 \wedge 10 = 00.$$

5. Метод кубітно-сигнатурного аналізу malware

Розглядається паралельний метод кубітного моделювання malware великих даних, який характеризується використанням сигнатурного аналізу і кубітними структурами даних, що дає можливість в паралельному режимі визначати належність поточного коду до існуючих деструктивних компонентів у malware library.

Мета сигнатурного аналізу полягає в суттєвому зменшенні (від квадратичної до лінійної) обчислювальної складності процедури унітарного кодування текстових (мови опису malware: assembler, C++, Java) значень атрибутів при обробці великих даних вхідного потоку, що містить тисячі malware.

Сутність сигнатурного аналізу полягає в швидкому перетворенні значення текстової змінної в код-сигнатуру, чисельне значення якої є унікальним з ймовірністю 0,9998, що дає підстави ідентифікувати сигнатурою текстові фрагменти malware. Далі вирішуються завдання:

1) Визначення приналежності значення текстової (строкової) змінної поточного malware до одного з уже існуючих закодованих значень шляхом порівняння сигнатур, коли створення нового унітарного коду не потрібно.

2) Підтвердження оригінальності значення строкової змінної, коли встановлений факт, що в результаті виконаного перегляду її сигнатура не збіглася з уже існуючими, отже, необхідно створення нового унітарного коду.

3) Створення таблиці за форматом <текст-сигнатура>, яку можна розглядати як <текст-

число>, дає можливість звести задачу унітарного кодування значень текстових змінних до обчислювальної складності 2n (замість 0,5n**2) і отримати таблицю <malware-сигнатура-код>. Тут синтез унітарних кодів пов'язаний з переглядом стовпчика всіх отриманих сигнатур, як десяткових чисел, представлених 16 двійковими розрядами, з метою заповнення одиничними значеннями координат апріорно нульового вектора покриття, розмірністю 2**16, за адресами, який формується сигнатурами текстів. Потім рахункова процедура (if U(i)=1 then j=j+1 and A(j)=i) перегляду всіх координат вектора покриття на предмет пошуку одиничних значень сформує вже компактний вектор координат, який буде представляти не що інше, як упорядкований вектор адрес A для унітарного кодування значень malware змінних. Наприклад, є множина десяткових цифр {5,2,1,9,7,4}, які необхідно впорядкувати на основі застосування процедури лінійної обчислювальної складності. Завдання вирішується шляхом розгляду множини цифр, як адрес, з метою подальшого занесення одиничних значень в координати вектора покриття за адресами-елементам: U=(110110101). Обчислювальна складність даної процедури складає n елементів множини. Далі послідовно виконується читання значень координат, де в разі фіксації одиниці в поточну комірку вектора значень записується адреса одиничної координати: if U(i)=1 then j=j+1 and A(j)=i. Результат обробки вектора U по останній процедурі має такий вигляд: A=(1,2,4,5,7,9). Обчислювальна складність процедури запису упорядкованих елементів практично дорівнює n елементів безлічі, якщо розкид значень чисел незначний, в межах одного порядку. Інтегрально дві процедури мають витрати, рівні Q=2n.

Слід зауважити, що двійковий вектор покриття може бути ефективно використаний для істотного підвищення продуктивності процедури кодування значень malware змінних.

Стан проблеми: для присвоєння коду-числа фрагменту тексту необхідно переконатися, що він є оригінальним у масштабах всього CSC-процесу. Це досягається шляхом порівняння фрагмента з усією бібліотекою вже закодованих текстів, що належать деякому атрибуту malware. Обчислювальна складність процедури тільки одного порівняння дорівнює Q=m*n, де m – довжина фрагмента, n – число рядків у бібліотеці закодованих malware.

Метод кубітно-сигнатурного аналізу malware на основі унітарного кодування пропонує: 1) Ідентифікувати текстові фрагменти malware кодами-сигнатурами. 2) Занести одиничні значення в координати вектора покриття V, адреси яких дорівнюють сигнатурам оригінальних malware U(S). 3) При аналізі вхідного потоку великої розмірності оригінальність malware по одному атрибуту перевіряється шляхом формування сигнатури S з подальшою перевіркою в координаті U(S)=1 значення одиниці. Якщо одиниця є, то текстовий фрагмент не є новим і виконується перехід до наступного malware. В іншому випадку, якщо U(S)=0, фрагменту присвоюється унітарний код-ідентифікатор, а сигнатура виступає адресою для занесення значення U(S)=1. Обчислювальна складність такої безперебірної процедури дорівнює Q=s+2, s – складність отримання сигнатури для рядка символів.

Приклад використання сигнатурного кодування текстових значень атрибутів представлений в таблиці:

Malware	S(M)	U[S(M)]=1 simulation
A	**73A1**	U(S1=0)=**1**000 0000
B	**FC67**	U(S2=0)=1000 **1**000
C	**2C67**	U(S2=0)=10**1**0 1000
D	**ACF0**	U(S3=0)=1010 1**1**00
E	**E143**	U(S4=0)=1001 110**1**
F	**FC67**	U(S2=1)=1001 1101
G	**ACF0**	U(S3=1)=1001 1101
H	**EF43**	U(S4=0)=1**1**01 1101

Результуючий U-вектор унітарного кодування практично без обчислювальних витрат дозволяє вирішувати такі завдання: 1) Визначати число одиничних координат, що дає можливість генерувати унітарні коди текстових значень змінних і довжину кожного коду. 2) Декодувати сигнатури і текстові фрагменти за координатами одиничних значень U-вектора. 3) Без перебірних обчислень визначати статус текстового фрагмента: є він новим чи вже існуючим в бібліотеці значень параметра.

Сигнатура виконує роль адреси-посередника між довгим текстовим malware фрагментом і вектором покриття, який моделює процес ідентифікації оригінальності значень текстових змінних: U[S(M)]={0,1}. Заповнення вектора покриття за правилом: U[S(M)]=1 дає можливість змоделювати кількість і якість оригінальних текстових фрагментів. У загальному випадку технологія моделювання невпорядкованих чисел для їх упорядкування за зростанням укладається в схему: Числа – Як адре-

си – Вектор одиниць – Запис адрес одиничних координат в порядку їх зростання. Технологія моделювання текстових фрагментів для їх упорядкування за зростанням сигнатур, що виконують роль адрес, укладається в схему: Тексти – Як адреси-сигнатури – Вектор одиниць – Запис адрес-сигнатур одиничних координат в порядку їх зростання.

Таким чином, інтерпретація чисел D=A, як адрес 1-координат для початкового 0-вектора, є ефективною технологією рішення комбінаторних завдань: U(D)=1. Слід врахувати, що malware вихідні тексти, в загальному випадку, неможливо уявити у вигляді адрес вектора. Тому необхідний буферний компонент, який, з одного боку, може компактно ідентифікувати malware змінну, а з іншого – бути її адресою. Таким компонентом може виступати 16-розрядна сигнатура (або hash-функція), як ідентифікатор-адреса для тексту і вектора U[S(M)]=1:

M	S(M)	U[S(M)]	D	U(D)	D*
A	**73A1**	**1**000 0000	3	00**1**0 00000	1
B	**FC67**	1000 **1**000	6	0010 0**1**000	2
C	**2C67**	10**1**0 1000	8	0010 0101**0**	3
D	**ACF0**	1010 11**0**0	2	0**1**10 01010	4
E	**E143**	1001 110**1**	4	011**1** 01010	6
F	**FC67**	1001 1101	9	0111 0101**1**	7
G	**ACF0**	1001 1101	1	**1**111 01011	8
H	**EF43**	**1**101 1101	7	1111 01**1**11	9

Big Data malware коди мають два суттєвих недоліки: 1) Для них необхідна велика пам'ять. 2) Для їх аналізу необхідно виконувати порівняння вхідного фрагмента даних з вже існуючими malware в бібліотеці або базі даних. Обчислювальна складність такого переборного порівняння дорівнює

Q=L(Input)xN(Records)xL(Record).

Чим більше записів і чим більша їх довжина, тим більше часу потрібно на визначення належності вхідного фрагмента до вже існуючого запису в бібліотеці. Якщо такого запису немає, то виконується поповнення бібліотеки новим записом після фіксації негативного результату порівняння вхідного фрагмента з бібліотечними атрибутами.

Як уникнути істотних витрат часу, що мають кубічну функціональну залежність обчислювальної складності? За рахунок надмірності пам'яті і додаткових обчислювальних процедур, що дають можливість зменшити обсяг прямих обчислювальних витрат, пов'язаних з перебором при порівнянні malware значення вхідної змінної з таблицею бібліотечних рішень. Для цього існує технологія хешування або сигнатурного стиснення двійкової інформації в 16-розрядний код сигнатуру-адресу, яка з імовірністю 0,9998 відображає як завгодно довгу двійкову послідовність. Платою за таку компактність є додаткові часові витрати для отримання сигнатури Q=L(Input), які можуть бути неприйнятними при сигнатурному стисненні невеликих обсягів даних, порядку сотень біт. Інтегрально обчислювальна складність сигнатурного методу аналізу вхідних потоків даних визначається виразом: Q = L(Input) + L(Sign) x N(Sign), де перший доданок формує час сигнатурного стиснення вхідної послідовності, другий – задає час порівняння вхідної сигнатури з бібліотечними аналогами. В результаті обчислювальна складність стає квадратичною, але вже по відношенню до сигнатурних кодів, які за розмірністю істотно менші, ніж вхідні потоки даних. Якщо порівнювати сигнатурні коди паралельно, що можна здійснити на низькорівневих мовах програмування або опису апаратури, то обчислювальна складність аналізу CSC-процесів стає лінійною і залежною від часу отримання сигнатури і числа бібліотечних аналогів, з якими виконується порівняння: Q=L(Input) + N(Sign).

Порівняння сигнатур здійснюється з метою привласнення кожному унікальному 16-розрядному коду десяткового номера, який далі може служити адресою для генерування унітарного коду сигнатури або вхідного потоку даних. Якщо число сигнатур в базі даних дорівнює 10, то це означає, що розмірність унітарного коду для ідентифікації сигнатур дорівнює 10.

Перетворення: <input – **Sign** – Digit – Ucode> має дизрапторну практичну спрямованість, яка пов'язана з відсутністю перебору на рівні даних при взаємодії вхідного потоку з бібліотекою або таблицею рішень. Однак такий перебір тут існує на рівні сигнатур, який дає можливість формувати компактну базу даних шляхом присвоєння кожній сигнатурі десяткового номера або унітарного коду для виконання паралельних логічних операцій:

M	S(M)	D[S(M)]	U{D[S(M)]}
A	**73A1**	1	**1**000 00
B	**FC67**	2	**11**00 00
C	**2C67**	3	**111**0 00
D	**ACF0**	4	**1111** 00
E	**E143**	5	**1111 1**0
F	**FC67**	2	**1111 1**0
G	**ACF0**	4	**1111 1**0
H	**EF43**	6	**1111 11**

Альтернативне рішення може бути отримано шляхом використання сигнатури як адреси для формування унітарного коду в масштабі 2**16=65536. Розмірність адресного простору дає можливість абсолютно без перебору і порівняння вирішувати завдання по визначенню належності вхідного потоку даних до значень атрибутів бібліотеки. Завдання вирішується шляхом запису одиничного значення за адресою-сигнатурою вхідного потоку. Це дає можливість звести процедуру ідентифікації вхідного потоку шляхом запису одиниці в нульову координату інтегрального вектора унітарних кодів. Якщо за даною адресою вже існує одиниця, то вхідний потік не є оригінальним і робиться висновок про його належність до значення конкретного атрибута.

M	S(M)	U[S(M)]	D[S(M)]
A	**73A1**	…01000 000…	1
B	**FC67**	…01100 000…	2
C	**2C67**	…01110 000…	3
D	**ACF0**	…01111 000…	4
E	**E143**	…01111 100…	5
F	**FC67**	…01111 100…	2
G	**ACF0**	…01111 100…	4
H	**EF43**	…01111 110…	6

Рішення даного завдання дозволяє: 1) Ідентифікувати великі потоки даних і вигляді компактної сигнатури (16-ковий алфавіт: 0-9, A, C, F, H, P, U), яка може являти собою адресу одиничного значення координати в унітарному векторному просторі, розмірністю 2**16; 2) Уникати кубічної складності переборного завдання з пошуку в базі даних вхідного потоку і привести її рішення до лінійної обчислювальної складності Q=2n. 3) Впорядкувати вхідні потоки по чисельним значенням сигнатур з метою зменшення обчислювальної складності при вирішенні завдань аналізу CSC-процесів. 4) Визначати актуальне число розрядів для унітарного кодування множини вхідних потоків даних після отримання множини оригінальних сигнатур.

Можна не використовувати буферний компонент у вигляді сигнатури, яку досить важко обчислювати через неможливість її генерувати паралельно. У цьому випадку пропонується кожному новому текстовому фрагменту malware привласнювати номер-ідентифікатор, який може виступати і адресою для формування унітарного коду фрагмента шляхом занесення одиниці за адресою у відповідному полі вектора покриття. Однак в даному випадку кожен новий текстовий фрагмент, що надходить на вхід <malware-address> кодера, потребує порівняння з вже існуючими текстами.

Формальна постановка задачі: отримати мінімальну множину символів з вхідного набору букв (тексту), що має повторення.

Example 1 (minimal set number should be ordered).

Input = {4,5,2,8,4,3,7,1,5,4}.

Buffer = (000 000 000). Writing 1-unit on addresses, pointed in Input-set:

4(000 100 000), 5(000 110 000), 2(010 110 000), 8(010 110 010), 4(010 110 010),

3(011 110 010), 7(011 110 110), 1(111 110 110), 5(111 110 110), 4(111 110 110).

Output = 1-unit address decoding (111 110 110) = {1,2,3,4,5,7,8}.

Computing Complexity Q=2n instead of Q=1/2[n**2].

Example 2 (minimal set number).

Input = {we are the world, we are the children}.

Buffer = (w-1, e-2, a-3, r-4, t-5, h-6, o-7, l-8, d-9, c-10, h-11, I-12 , n-13).

Initial U-vector = (000 000 000 00) – Resulting U-vector = (111 111 111 11)

Output = {w, e, a, r, t, h, o, l, d, c, n}

Example 3 (minimal set number).

1) Input = {we are the world, we are the children}.

2) Buffer (coder) = (a-1 b-2 c-3 d-4 e-5 f-6 g-7 h-8 i-9 j-10 k-11 l-12 m-13 n-14 o-15 p-16 q-17 r-18 s-19 t-20 u-21 v-22 w-23 x-24 y-25 z-26).

3) Initial U-vector = (000 000 000 000 000 000 000 000 00).

4) Resulting U-vector = (101 110 010 001 011 001 010 010 00).

5) Output (pattern) = {a, c, d, e, h, l, n, o, r, t, w}.

6. Модель кубітно-матричного процесора для CSC-компьютинга

Рішення задач malware аналіза пов'язано зі створенням моделі, методу та інтерпретативного гнучкого процесора CSC-комп'ютинга, спрямованого на автоматичний синтез і аналіз кубітних логічних схем, орієнтованих на моніторинг, моделювання, розпізнавання і управління CSC-процесами і явищами:

1) Процесор CSC-комп'ютинга на основі синтезу кубітних логічних схем для моніторингу, моделювання, розпізнавання і управління CSC-процесами.

2) Кубітно-матрична модель універсуму змінних і універсуму значень кожної багатозначної змінної для синтезу логічних схем

malware-паттернів, орієнтованих на аналіз і управління CSC-процесами.

3) Кубітно-матричний метод синтезу логічної схеми для моделювання та розпізнавання malware-функціональностей на основі унітарного кодування значень багатозначних змінних з метою оптимального управління CSC-процесом.

4) Кубітно-матричний інтерпретативний метод моделювання вхідних потоків malware-даних CSC-процесів на основі використання еталонних логічних елементів malware-функціональностей з унітарним кодуванням багатозначних змінних. Метод відрізняється від розумних блокчейн-контрактів, що представляють собою виконуваний код, інтерпретативним аналізом гнучких структур даних, що дає можливість змінювати модель CSC-процесу в online режимі реального часу.

5) Тестування і верифікація кубітно-матричної архітектури CSC-комп'ютинга на прикладах аналізу і розпізнавання процесів у вхідних потоках великих даних, пов'язаних з кіберфізичним відображенням і управлінням діяльністю організації.

Кубітно-матрична інтерпретативна модель malware-паттернів і її унітарно-кодований екземпляр для моделювання і актюаторного управління CSC-процесом представлені на рис. 10.

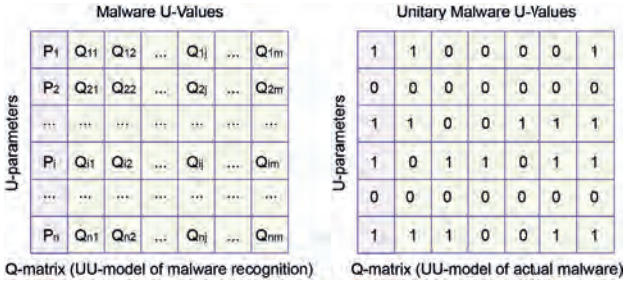

Рис. 10. Кубітна UU-матриця CSC-процесу

Фактично, Q-матриця являє собою формальний запис протоколу, анкети або ідеального блокчейн-контракту, який додатково виробляє актюаторні сигнали, що спонукають систему виконувати певні дії для досягнення мети – ліквідації malware. Таким чином, прямі і безпосередні інтеракції між CSC-роботом-автоматом і користувачем створюють CSC-комп'ютинг, який використовує Q-матрицю і логічні схеми, спрямовані на отримання CSC-сервісів за мінімальний час без участі людини.

7. Сигнатурно-кубітний метод синтезу еталонних логічних схем

Комбінаційна модель для сигнатурно-кубітного методу синтезу еталонних логічних схем malware-функціональностей використовує унітарне кодування сигнатур для кодів деструктивних компонентів і формування кубітних матриць, що дає можливість в паралельному режимі визначати наявність malware в обчислювальній системі або мережі. Щоб показати схему для формування структур даних malware-функціональності (U, Q, X), необхідно з'єднати вхідний потік даних S для кожної malware-змінної. Схема для формування структур даних CSC-процесу (V, P, Y) з'єднує вхідні потоки даних для змінних F щодо кожної malware-функціональності (рис. 11).

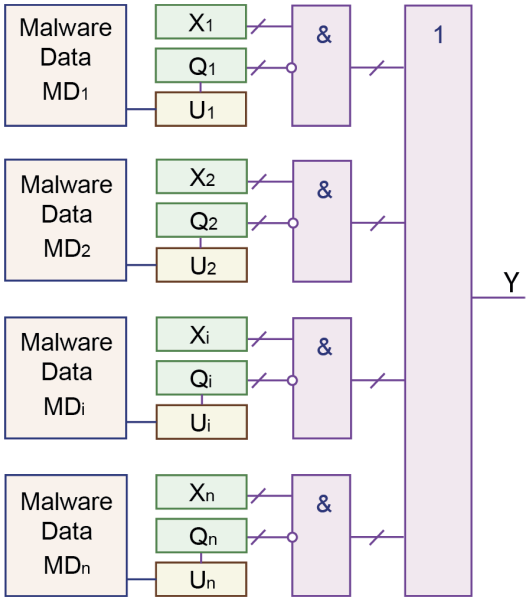

Рис. 11. Сигнатурно-матрична модель комбінаційної схеми для malware-аналізу

Універсум універсумів (UU) являє собою дворівневу модель процесу або явища, призначену для активного цифрового моделювання з метою точного розпізнавання великих потоків вхідних даних шляхом метричного порівняння з заданими еталонами. Кожен з параметрів представлений значеннями, які в сукупності складають універсум другого рівня. В результаті виходить UU-метрика функціональності еталонного процесу або явища, щодо якого можна вимірювати-моделювати потік вхідних даних, які претендують на дану роль. Для проектування CSC-комп'ютинга використовуються два види моделей логічних процесорів. Кубітно-регістрова модель (КРМ) оперує двома входами логічного and-елемента, на які подаються:

1) кубітний вектор-еталон багатозначної змінної malware-функціональності; 2) вхідний вектор даних CSC-процесу, що підлягає розпізнаванню. Обидва вектора паралельно порівнюються між собою за процедурою Y=∨[(X∧Q)⊕X], яка включає три паралельних операції, що призводить до двійкового результату: Y = 0, якщо X належить вектору Q, в іншому випадку Y=1. Кубітно-логічна модель (КЛМ) оперує одним входом логічного and-елемента, на який подається значення-адреса з даних CSC-процесу, що відповідає цій змінній. За адресою, отриманою в результаті кодування текстового фрагмента, визначається координата кубітного вектор-еталона багатозначної змінної malware-функціональності, значення якої формує стан виходу Y для розпізнавання значення вхідних даних щодо заданого кубітного вектора, що формує логічний еталон. Іншими словами, одна ітерація моделювання на кубітному елементі КЛМ-моделі malware-функціональності визначає належність вхідного значення до кубітного вектора логічного еталона. КРМ модель змінної malware-функціональності дає можливість паралельно, за один автоматний цикл, визначати належність вхідного вектора до кубітного вектора логічного еталона. У цьому їх відмінність. Природно, що вхідні впливу для обох моделей формуються з текстових фрагментів вхідних даних шляхом їх унітарного кодування на універсальній множині значень кожної змінної malware-функціональності.

8. Процесорна архітектура CSC-комп'ютинга. Проблема і рішення

Проблема CSC комп'ютинга полягає в складності формалізації malware-логіки, яку необхідно привести до цифрового детермінізму функціональних відповідностей, що виключає ймовірність і невизначеність. Піти від евристики у бік автоматизації синтезу та аналізу логічних схем CSC-процесингу для моделювання та передбачення malware-колізій – завдання, актуальне для ринку. Для його вирішення пропонується трансформувати функціонально закінчений потік великих даних до структурованої матричної двійкової форми, яка унітарно кодує сукупність malware-змінних і всі можливі їх значення, складові універсуму примітивів. Метод для вирішення проблеми представлений синтезом класів еквівалентних відношень на сукупності змінних P:

$$P = \{P_1, P_2, \dots, P_i, \dots, P_j, \dots, P_n\}, P_i \cap P_j = \emptyset,$$

де кожна з них

$$P_i = \{P_{i1}, P_{i2}, \dots, P_{ik}, \dots, P_{ir}, \dots, P_{in}\}$$

приймає універсальну множину malware-значень, що створюють між собою еквівалентні відношення $P_{ik} \sim P_{ir}$ і утворюють при цьому порожні перетини $P_{ik} \cap P_{ir} = \emptyset$.

Метрика: визначення, аксіоми і рівняння CSC-комп'ютинга.

1) Метрика є спосіб вимірювання відстаней $d_i \in D$ між процесами і явищами в просторі заданих параметрів з виконанням аксіоми циклічного конволюційного замикання:

$$D = \sum_{i=1}^{n} d_i = 0 \quad .$$

2) Вимірювання – процедура визначення відстані між кінцевою множиною процесів або явищ, відмінних від нуля.

3) Параметрами виступають логічні змінні $P = \{P_1, P_2, \dots, P_i, \dots, P_n\}$, які із заданим ступенем адекватності описують процес або явище.

4) Змінні визначаються за допомогою їх значень $P_i = \{P_{i1}, P_{i2}, \dots, P_{ij}, \dots, P_{in}\}$, кількість яких не може бути менше двох.

5) Матриця універсумів є упорядкована сукупність змінних і їх значень $U = [P_{ij}] = [U_{ij}]$ для метричного вимірювання процесу або явища. Унітарно кодована матриця U має одиничні значення по всіх координатах. Матриця універсумів U може бути представлена одним вектором шляхом конкатенації її рядків:

$$P = (P_1 * P_2 * \dots * P_i * \dots * P_n).$$

6) Матриця malware $Q = [Q_{ij}]$ є підмножина значень змінних універсальної матриці $Q \in U$, яка формує зразок конкретного процесу або явища. Найбільш зручною формою завдання malware є матриця унітарного двійкового кодування значень змінних.

7) Матриця вхідних даних $X = [X_{ij}]$ є підмножина (двійкових) значень змінних універсальної матриці $X \in U$, яка формує фрагмент реального процесу або явища.

8) Матриця вимірювання $Y = [Y_{ij}] = [Q_{ij}] \oplus [X_{ij}]$ є підмножина $Y \in U$ (двійкових) значень змінних універсальної матриці, яке формує відмінності в однойменних координатах матриць Q і X шляхом виконання паралельної xor-операції між malware і фрагментом обчислювального процесу або явища.

9) Відстань є скалярна оцінка кількості відмінностей, зазначених одиницями, в однойменних координатах матриць вимірюваних процесів або явищ. Відстань визначається шляхом підрахунку одиничних координат в матриці вимірювання:

$$d\,(Q,X) = \sum_{j=1,m}^{i=1,n} Y_{ij}.$$

10) Функція відмінності є відношення відстані (кількості відмінностей в однойменних координатах) до загальної кількості координат матриці (вимірювання):

$$\mu = \frac{d\,(Q,X)}{n \times m} = \frac{\sum_{j=1,m}^{i=1,n} Y_{ij}}{n \times m}.$$

Далі розглядається процесорна архітектура активного online кіберфізичного cyber security комп'ютинга, яка характеризується моніторингом вхідних потоків malware-даних, їх подальшим моделюванням на еталонних логічних схемах malware-функціональностей, що дає можливість в online режимі актюаторно управляти процесом видалення деструктивних компонентів.

Логічна структура, представлена на рис. 12, характеризується комп'ютинговою архітектурою malware-аналітики, яка має всі вісім компонентів моделі універсального обчислювача, взаємодіючих між собою за формулою: вхід R ініціює команди для виконання CSC-процесу, який починається з отримання D-ресурсів: фінансових, кадрових, інформаційних. Вони надходять на виконавчий механізм E, який активує сенсори, що передають інформацію по шині моніторингу M для подальшого формування потоку S-даних, призначеного для синтезу U-матриці універсуму змінних для опису CSC-процесу за допомогою універсумів вербальних значень кожної змінної. Вона слугує базовим форматом для синтезу X-матриці векторів вхідних даних по кожній змінній для виконання ітерації моделювання відносно Q-матриці, що задає сукупність кубітних двійкових векторів для опису еталонного malware-паттерна за всіма змінними. Це дає можливість на основі &-операції обчислювати функцію належності μ, що має вихід візуалізації стану V CSC-процесу, і відповідну Y у вигляді чисельного значення кількості різних двійкових однойменних координат, отриманого при порівнянні матриць X і Q, що визначається за допомогою паралельної операції (X ∧ notQ) = Y. Це дає можливість синтезувати актюаторні вербальні сигнали W, відповідні одиничним значенням координат вихідної матриці Y в форматі U-матриці, які по шині A ініціюють обчислювальні процедури, спрямовані на усунення розходжень між X-матрицею і еталонною Q-матрицею malware-паттерну шляхом корекції координат X-матриці за допомогою інфраструктури E, що забезпечує виконання CSC-процесу компанії для отримання сервісів або продукції P.

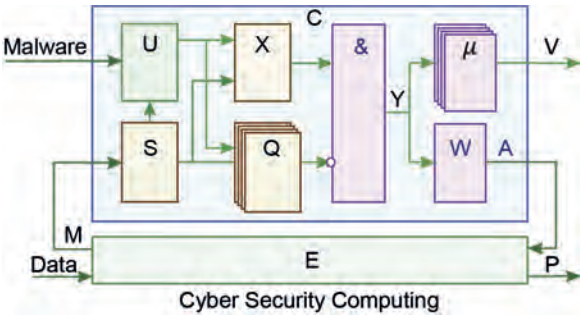

Рис. 12. Логічна матрична паралельна схема CSC-комп'ютинга

Архітектура CSC-комп'ютинга характеризується кінцевою множиною Q-матриць для паралельної обробки вхідних X-матриці з метою отримання μ-матриць. Вона також обчислюються паралельно, що представляє собою універсум функцій приладдя чергового вхідного впливу malware-паттернам, серед яких вибирається мінімальна оцінка відмінностей і скалярні оцінки для усунення malware.

Оригінальність моделі кубітно-матричного процесора, що відповідає архітектурі, представленої на рис. 12, характеризується використанням кубітних матриць даних для виконання операцій CSC-комп'ютинга, його структура містить механізми управління C і виконання E, з'єднані шинами моніторингу M і актюації A, де блок управління має зовнішній вхід команд R і вихід візуалізації стану CSC-процесу V. Блок виконання містить зовнішній вхід даних D і вихід сервісів або продукції P, що в сукупності становить вісім компонентів, які характеризують CSC-комп'ютинг. Він починається з визначення вхідного потоку даних S, що надходить від сенсорів виконавчого механізму, обслуговуючого CSC-процес, який призначений для синтезу U-матриці універсуму змінних і опису CSC-процесу за допомогою універсумів вербальних значень кожної змінної. Вона є базовим шаблоном для синтезу X-матриці вхідних даних і наступного моделювання сумісно з вже визначеними Q-матрицями, що задають множину еталонних malware-паттернів. Це дає можливість на основі &-операції паралельно обчислювати n матриць Yi = (X ∧ notQi), (i=1,n) моделювання, аналіз яких на мінімальне число одиничних координат min(Yi=1), (i=1,n), дозволяє визначити номер і матриці, що має мінімальне значення з n функцій належностей

$$\mu = \min_i \mu_i \leftarrow \mu_i = \sum_{j=1,k}^{r=1,m} Y_{ijr}.$$

Вони формують чисельні значення відмінностей двійкових однойменних координат, отриманих при порівнянні матриці X і n матриць Q. Це дає можливість синтезувати матрицю актюаторних вербальних сигналів W, відповідну одиничним значенням координат вихідної матриці Y, що має minμ, в форматі вербальних значень U-матриці, які ініціюють обчислювальні процедури у виконавчому механізмі E, спрямовані на усунення розходжень між X-матрицею і Q-матрицею malware-паттерну з min μ, шляхом корекції координат X-матриці, що забезпечує оптимальне виконання мети P CSC-процесу.

Алгоритм роботи кубітно-матричного процесора, представленого на рис. 13, визначає послідовність процедур комп'ютинга в часі і просторі для розпізнавання malware-паттернів у вхідних потоках даних і вироблення актюаторних сигналів-впливів з метою усунення розбіжностей між ними або деструкції malware, що починається з блоку 1) ініціювання алгоритму з наступним переходом до 2) блоку смислового аналізу вхідних даних S, які надходять від входу D і сенсорів M цифрової інфраструктури. При цьому виконується 3) перевірка наявності U-matrix з метою переходу на блок 6, а в разі її відсутності алгоритм активує процедура розбиття текстових фрагментів на непересічні класи еквівалентності для 4) вилучення універсуму malware або змінних-примітивів, де для кожної з них визначається універсум значень. Це в сукупності формує U-матрицю універсуму універсумів, яка служить шаблоном для наступного 5) синтезу кубітних Q-матриць malware-паттернів і 6) кубітної X-матриці вхідних даних, які у подальшому використовуються для виконання 7) паралельної &-операції та отримання n матриць Yi=(X∧notQi), (i=1,n) моделювання, в яких підраховується кількість одиничних координат min (Yi = 1), (i = 1, n). Це дає можливість 8) визначити номер і матриці, що має мінімальне значення з n функцій належностей

$$\mu = \min_i \mu_i \leftarrow \mu_i = \sum_{j=1,k}^{r=1,m} Y_{ijr},$$

які формують чисельні значення відмінностей двійкових однойменних координат, отриманих при порівнянні матриці X і n матриць Q. Це дозволяє 9) синтезувати матрицю актюаторних вербальних сигналів W, відповідну одиничним значенням координат вихідної матриці Y, що має minμ, в форматі вербальних значень U-матриці, які 10) при μ≠0 ініціюють 11) обчислювальні процедури у виконавчому механізмі E, спрямовані на усунення розходжень між X-матрицею і Q-матрицею malware-паттерну з minμ, шляхом корекції координат X-матриці, що забезпечує оптимальне виконання мети P CSC-процесу. При існуванні μ≠0 виконується завершення роботи алгоритму на заданій матриці вхідних впливів, який забезпечує реалізацію CSC-комп'ютинга, орієнтованого на деструкцію malware, за рахунок цифровізації, автоматизації та просторово-часової оптимізації CSC-процесу при створенні сервісу або продукту.

Таким чином, матрична структура даних для паралельного виконання операцій CSC-комп'ютинга дає можливість істотно підвищити швидкодію інтерпретативного паралельного моделювання вхідних потоків malware-даних для розпізнавання в них malware-паттернів з метою генерування актюаторних сигналів, що усувають malware або відмінності в X-матриці по відношенню до одного з еталонних CSC-паттернів.

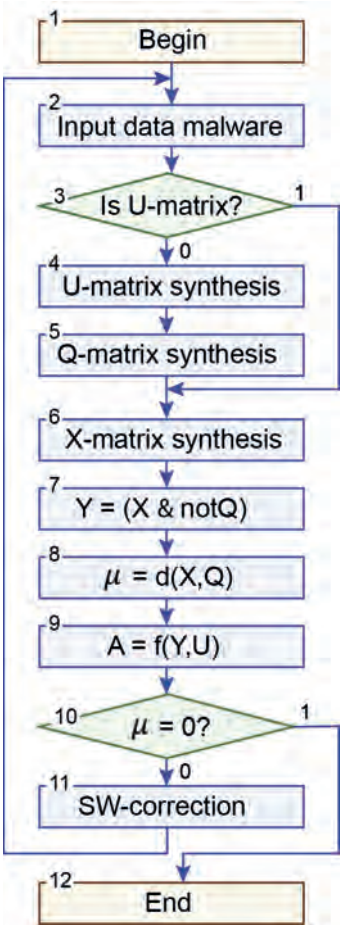

Рис. 13. Алгоритм CSC-комп'ютинга

8. Висновки

Наукова новизна полягає в створенні логічного процесора для паралельного моделювання та розпізнавання malware-паттернів в потоках великих даних на основі створення інтерпретативних кубітних матричних моделей, методів і архітектур CSC-комп'ютинга, спрямованого на автоматичний синтез і аналіз логічних схем, орієнтованих на моніторинг і управління CSC-процесами і явищами для усунення деструктивних компонентів у кіберпросторі:

1) Запропоновано паралельний сигнатурно-кубітний метод моделювання malware-driven великих даних, який характеризується використанням сигнатурного аналізу і кубітними структурами даних, що дає можливість в паралельному режимі визначати належність поточного коду до існуючих деструктивних компонентів у malware library.

2) Розроблено сигнатурно-кубітний метод синтезу еталонних логічних схем malware-функціональностей, який відрізняється від аналогів унітарним кодуванням сигнатур для кодів деструктивних компонентів і формуванням кубітних матриць для моделювання, що дає можливість в паралельному режимі визначати наявність malware в обчислювальній системі або мережі.

3) Розроблено процесорну сигнатурно-кубітну модель-архітектуру активного online cyber security комп'ютинга, яка характеризується моніторингом вхідних потоків malware-даних, їх подальшим моделюванням на еталонних логічних схемах malware-функціональностей, що дає можливість в online режимі актюаторно управляти процесом видалення деструктивних компонентів.

4) Напрями майбутніх досліджень. Так само як біологічні віруси деструктурують людину, кібервіруси вражають організм world-комп'ютинга в масштабах планети, завдаючи людству багатомільярдну матеріальну та моральну шкоду. Одним з можливих варіантів може бути кіберімунітет, як кіберфізичний моральний комп'ютинг метричного вичерпного моніторингу всіх процесів і явищ для цифрового human-free управління на основі моделювання, передбачення наслідків від malware, діагностування та знищення некорисних програмних додатків.

Література:

1. Lehto, Martti, Neittaanmäki, Pekka Cyber Security: Analytics, Technology and Automation, Springer, 2015. 269 p.]
2. Orojloo Hamed, Mohammad Abdollahi Azgomi. Modelling and evaluation of the security of cyber-physical systems using stochastic Petri nets. IET Cyber-Physical Systems: Theory & Applications (2019), 4 (1), P. 50-57.]
3. https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/
4. https://www.gartner.com/doc/3891569?src Id=1-7251599992&cm_sp=swg-_-gi-_-dynamic
5. *Gupta A.and Jha RK*, "A Survey of 5G Network: Architecture and Emerging Technologies," in IEEE Access. Vol. 3. 2015.P. 1206-1232.
6. *Zhu C., Leung VCM, Shu L. and ECH Ngai*, "Green Internet of Things for Smart World," in IEEE Access, Vol. 3. 2015. P. 2151-2162.
7. *Christidis K. and Devetsikiotis M.*, "Blockchains and Smart Contracts for the Internet of Things," in IEEE Access. Vol. 4. 2016. P. 2292-2303.
8. Blockchains: How They Work and Why They'll Change the World IEEE Spectrum. October 2017.
9. *Zanella A., Bui N., Castellani A., Vangelista L. and Zorzi M.*, "Internet of Things for Smart Cities," in IEEE IoT Journalю Vol. 1, no. 1. Feb. 2014.P. 22-32.
10. *Frahim J.* Securing the Internet of Things: A Proposed Framework / J. Frahim // Cisco White Paper.- 2015.
11. *Kharchenko V.* Green IT Engineering: Concepts, Models, Complex Systems Architectures / V. Kharchenko, Y. Kondratenko, J. Kacprzyk (Eds.) // In the book series "Studies in Systems, Decision and Control" (SSDC). Vol. 1. 2017.Berlin, Heidelberg: Springer International Publishing.
12. *Kharchenko V.* Green IT Engineering: Components, Networks and Systems Implementation / V. Kharchenko, Y. Kondratenko, J. Kacprzyk (Eds.) // In the book series "Studies in Systems, Decision and Control" (SSDC). 2017.Vol. 2. Berlin, Heidelberg: Springer International Publishing.
13. Memory-Driven Computing. [Online]. Available: https://www.labs.hpe.com/next-next/mdc
14. Benenti G., Casati G., Strini G. Principles of Quantum Computation and Information. Volume 1: Basic Concepts.-World Scientific.- 2004.- 256 p.
15. Imai Hiroshi, Hayashi Masahito. Quantum Computation and Information. From Theory to Experiment.- Springer. 2006. 234 p.
16. *Nielsen MA, Chuang IL* Quantum Computation and Quantum Information. Cambridge University Press. 2010. 710 p.
17. *Abramovici M.* Digital System Testing and Testable Design / M. Abramovici, MA Breuer and AD Friedman. Comp. Sc. Press. 1998. 652 p.
18. *Benso A.* Control-flow checking via regular expressions / A. Benso, S. Di Carlo, G. Di Natale, P. Prinetto, L. Tagliaferri // Proceedings 10th Asian Int. Test Sym-

posium. Kyoto. 2001. P. 299-303. [Online]. Available: http://dl.acm.org/citation.cfm?id= 872025.872649
19. *Vladimir Hahanov.* Cyber Physical Computing for IoT-driven Services. New York. Springer. 2018. 279p.
20. *Hahanov V.I.* Qubit technologies for analysis and diagnosis of digital devices / V.I. Hahanov, T. Bani Amer, S.V. Chumachenko, E.I. Litvinova // Electronic Modeling. Vol. 37, no. 3. 2015. P. 17-40.
21. *Хаханов В.И.* Кубитные структуры данных вычислительных устройств / В. И. Хаханов, В. Гариби, Е. И. Литвинова, А. С. Шкиль // Электронное моделирование. - 2015. - Т. 37, № 1. - С. 76-99.
22. *Hahanov V.* Cloud-driven Cyber Managing Resources / V. Hahanov, S. Chumachenko, E. Litvinova, O. Mishchenko, I. Yemelyanov, Bani Amer Tamer // Australian Journal of Scientific Reseach. № 1(5). 2014. С. 202-215.
23. *Hahanov I.* QuaSim – Cloud Service for Digital Circuits Simulation / I. Hahanov, W. Gharibi, I. Iemelianov, T. Bani Amer // Proceedings of IEEE East-West Design & Test Symposium. 2016. Yerevan, Armenia. P. 363-370.

Надійшла до редколегії 23.12.2018
Рецензент: д-р техн. наук, проф. Дрозд О.В.

Адамов Олександр Семенович, старший викладач кафедри АПОТ ХНУРЕ. Наукові інтереси: кібербезпека. Адреса: Україна, 61166, Харків, пр. Науки, 14, тел. 70-21-326. E-mail: oleksandr.adamov@nure.ua.

Хаханов Володимир Іванович, д-р техн. наук, професор кафедри АПОТ ХНУРЕ. Наукові інтереси: технічна діагностика цифрових систем, мереж і програмних продуктів. Захоплення: баскетбол, футбол, гірські лижі. Адреса: Україна, 61166, Харків, пр. Науки, 14, тел. 70-21-326. E-mail: hahanov@nure.ua.

Adamov Aleksandr Semenovich, Senior Lecturer, Design Automation Department, NURE. Scientific interests: cybersecurity. Address: Ukraine, 61166, Kharkiv, Nauki Avenue, 14, tel. 70-21-326. E-mail: oleksandr.adamov@nure.ua.

Hahanov Vladimir Ivanovich, Dr. Tech. Sciences, professor, Design Automation Department, NURE. Scientific interests: technical diagnosis of digital systems, networks and software products. Hobbies: basketball, football, mountain skiing. Address: Ukraine, 61166, Kharkiv, Nauki Avenue, 14, tel. 70-21-326. E-mail: hahanov@nure.ua.