УДК 004.63

РОЗШИРЕНИЙ МЕТОД АНАЛІЗУ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З МЕТОЮ СТВОРЕННЯ СИГНАТУР

DOI 10.36994/2707-4110-2020-1-28-14

Єгоров С.В., к.т.н., Національний авіаційний університет, Київ, Україна.
sehorov@gmail.com

Шкварницька Т.Ю., к.т.н., доц., Національний авіаційний університет, Київ, Україна.
tskvarnitska@ukr.net

Анотація. Розглянуто метод базового статичного аналізу шкідливого програмного забезпечення, що базується на пошуку та аналізі строк в файлах, що побудовані з використанням формату PE (Portable Executable). Розглянуто метод базового статичного аналізу шкідливого програмного забезпечення, що базується на аналізі заголовків файлів, що виконуються та динамічних бібліотек, які побудовані з використанням формату PE. Розглянуто розширений метод статичного аналізу, який крім аналізу строк та заголовків файлів використовує дизасемблювання файлів, що виконуються та динамічних бібліотек і подальший аналіз отриманого асемблерного тексту.

З метою проникнення в операційну систему, зловмисники використовують спеціалізоване програмне забезпечення та мережеві атаки. Причому мережева атака не обов'язково повинна бути масивною та широкомасштабною. Для проникнення в окрему взятую операційну систему, наприклад, можна скористатись уразливостями як самої операційної системи так і програмного забезпечення, яке встановлено на такій операційній системі. Причому успішні атаки такого типу часто робляться тихими та малопомітними.

Для запобігання хакерським атакам які супроводжуються прихованим встановленням програмного забезпечення та мінімізації шкоди від таких атак необхідно вчасно вживати адекватні контрзаходи. Одним із самих розповсюджених та легких методів боротьби з хакерами це своєчасне оновлення програмного забезпечення, вірусних баз, інсталяція та налаштування брандмауера.

Все що пов'язано з оновленням програмного забезпечення – це реакція на вже виявлені загрози. Тому оновлення програмного забезпечення на надає захист від щойно виявлених загроз. Саме тому сигнатури вірусних баз створюються в результаті аналізу виявлених шкідливих програм. Для підсилення виявлення шкідливого програмного забезпечення також антивірусними програмами використовується аналіз поведінки програм. Але навіть у цьому випадку необхідно аналізувати дизасембльований текст шкідливих програм для виявлення нових типів аномальної активності. Тому аналіз шкідливого програмного забезпечення є задачею актуальною та визначає напрям дослідження.

Ключові слова: Кібербезпека, інформаційна безпека, статичний аналіз, дизасемблювання, зворотна розробка.

ADVANCED METHOD OF ANALYSIS OF MALICIOUS SOFTWARE FOR THE PURPOSE OF CREATING SIGNATURES

Serhii Yehorov, Ph.D., National Aviation University, Kyiv, Ukraine.
sehorov@gmail.com

Tetyana Shkvarnytska, Ph.D., Ass. Prof., National Aviation University, Kyiv, Ukraine.
tskvarnitska@ukr.net

Abstract. *The method of basic static analysis of harmful software is considered, which is based on searching and analyzing the term in files that are built using the PE (Portable Executable) format. The method of basic static analysis of malicious software is considered, which is based on the analysis of headers of executable files, and dynamic libraries, which are built using the PE format. An extended static analysis method is considered, which, in addition to analyzing the term and file headers, uses disassembly of executable files and dynamic libraries and further analysis of the resulting assembler text.*

In order to penetrate the operating system, cybercriminals use specialized software and network attacks. Moreover, a network attack does not have to be massive and widespread. To penetrate a particular operating system, for example, you can take advantage of vulnerabilities in both the operating system itself and the software that is installed on such an operating system. Moreover, successful attacks of this type are often made quiet and unobtrusive.

To prevent hacker attacks that are accompanied by hidden software installation and to minimize harm from such attacks, it is necessary to apply adequate countermeasures in a timely manner. One of the most widespread and easy methods of fighting hackers is the timely updating of software, virus databases, installation and configuration of a firewall.

Everything related to software updates is a reaction to threats that have already been identified. Therefore, the software update does not provide protection against the threats just identified. That is why the signatures of virus databases are created as a result of the analysis of the detected virus programs. Antivirus software also uses program behavior analysis to enhance the detection of malware. But even in this case, it is necessary to analyze the disassembled text of malicious software to identify new types of abnormal activity. Therefore, the analysis of malicious software is an urgent task and determines the direction of the study.

Keywords: *Cybersecurity, information security, static analysis, disassembly, reverse engineering.*

Вступ

Одним із самих розповсюджених та легких методів боротьби з хакерами це своєчасне оновлення програмного забезпечення, вірусних баз, інсталяція та налаштування брандмауера.

Все що пов'язано з оновленням програмного забезпечення – це реакція на вже виявлені загрози. Тому оновлення програмного забезпечення на надає захист від щойно виявлених загроз.

Єдиним ефективним методом виявлення нових загроз та створення ефективних локальних та мережових сигнатур це аналіз щойно виявленого шкідливого програмного забезпечення.

Тому аналіз шкідливого програмного забезпечення є задачею актуальною та визначає напрям дослідження.

В [1] розроблено метод оцінки зрілості екосистем але не розглянуто оцінку шкідливого програмного забезпечення, яке здатне проникнути в цю екосистему. В результаті заражена вірусною програмою екосистема буде працювати неналежним чином, або на користь зловмисника.

В [2] Побудована математична модель кіберзахисних дій, що дозволяє оцінити вартість кіберзахисту але не розглянуті методи забезпечення кіберзахисту.

В [3] Були розглянуті технології для поліпшення безпеки польотної зони. Але автор и [3] не врахували той факт, що програмні системи керування можуть контролюватися зловмисником. Для ефективного усунення зламу необхідно буде аналізувати код програми, за допомогою якої зламник контролює систему.

В [4] проблеми проектування операційних систем для керування безпілотними літальними апаратами. Але не достатньо розглянуто проблему захисту від кібератак.

В роботі [5] представлена розглянути проблеми діагностування безпілотних літальних апаратів, але не достатньо розглянуті питання кібербезпеки.

В роботі [6] розглянуто захист банківських систем за допомогою симетричного криптографічного алгоритму, але перехоплювати дані можна, при певних умовах до шифрування і без асиметричних алгоритмів захист симетричними алгоритмами шифрування не завжди ефективний.

В роботі [7] розглянуто захист каналу зв'язку з безпілотним літальним апаратом методом обфускації. але дані можна перехопити до обфускації. Без детального аналізу шкідливого програмного забезпечення локалізувати канал витоку буде не можливо.

В роботі [8] розглянуто методи біометричної аутентифікації. Але якщо операційна система має відповідне шкідливе програмне забезпечення, то ці методи можна буде обійти.

В роботі [9] розглянуто алгоритм захисту від квантових комп'ютерних атак. Але враховуючи ціни на квантові комп'ютери така атака буде занадто дорогою вартісною.

В роботі [10] запропоновано новий алгоритм обміну криптографічними ключами. Але без належного захисту операційної системи цей криптографічний алгоритм можна обійти.

В роботі [11] запропонований метод аудиту кібербезпеки не забезпечує аналізу шкідливого програмного забезпечення.

В роботі [12], використовуючи передову технологію software-defined networking (SDN), було розроблено метод Moving target defense (MTD) на основі графа атаки, який перемішує мережеві конфігурації хоста (наприклад, MAC / IP / адреси портів) на основі його критичності, що дуже корисно для зловмисників, коли хост знаходиться на шляху (атаках) атаки.

Аналіз та пошук строк

Строка в програмі являє собою деяку послідовність символів. Наприклад send. Якщо програма копіює будь-яку інформацію, звертається за url або IP адресою, виводить сповіщення, створює файл це ознаки того, що програма містить строки.

Здійснюючи пошук строк в програмі можна скласти уявлення про загальний функціонал програми. Припустимо, якщо програма звертається за url або IP адресою, то в програмі можна знайти відповідні строки, що містять url, IP адресу, номер порта. Строки зберігаються в програмі в форматі ASCII, або Unicode та закінчуються нульовим символом, що вказує на кінець строки.

Існує безліч програм, які здатні шукати строки в програмах. Наприклад, Strings з комплекту програм Марка Русиновича. Ця програма дозволяє шукати строки в файлах будь-якого типу ігноруючи при цьому контекст та форматування. Але при цьому може виявити строки, що такими не є.

На рис. 1 показано уривок результату пошуку строк в шкідливій програмі, що була виявлена на комп'ютері.

Строка (1) являє собою формат для текстового сповіщення. Строки (2), (3) є портом та IP адресою відповідно. IP адреса була змінена навмисно для безпечного запуску програми. Інформація, що міститься в строках (4) – (9) нашоухує не мисль, що цей файл є кейлоггером, що передає зібрану інформацію за зазначеною IP адресою. Це припущення підтверджується інформацією, що міститься в строках (10) - (14).

SetWindowsHookExA, що займає строку (10) встановлює визначену програмою процедуру перехоплення сповіщень в ланцюжок перехоплень. Може використовуватись для моніторингу системи щодо певних типів подій. Ці події пов'язані або з певним потоком, або з усіма потоками на одному робочому столі, з якого був викликаний потік. Буква «А» у кінці API функції означає, що ця функція працює зі строками ASCII. CallNextHookEx, що займає позицію (11) передає інформацію про перехоплення до наступної процедури перехоплення в поточному ланцюжку перехоплень.

```
tN3
tC@
%X0@
%\0@
@@
p@@
%\0@
[Window: %s - at %s]          (1)
1337                          (2)
127.0.0.1                      (3)
[BACKSPACE]                   (4)
[ENTER]                        (5)
[SPACE]                        (6)
```

[TAB]	(7)
[SHIFT]	(8)
[CONTROL]	(9)
SetWindowsHookExA	(10)
CallNextHookEx	(11)
GetMessageA	
GetKeyState	(12)
USER32.dll	(13)
WS2_32.dll	(14)

Рис. 1. Уривок результату пошуку строк в шкідливій програмі

Процедура перехоплення може викликати цю функцію до або після обробки інформації про перехоплення. `GetKeyState`, що розташована в позиції (12) отримує статус вказаного віртуального ключа. Статус визначає те, що клавіша натиснута, не натиснута або перемикається (по черзі при кожному натисканні клавіші натискається і відпускається). Бібліотека `USER32.dll` містить елементи графічного інтерфейсу користувача та `WS2_32.dll` містить функції роботи з мережею.

Аналіз структури файлів

Кожний файл, що виконується (а також бібліотеки) в Windows. Мають певний формат. Заголовок файлу PE (Portable Executable) складається із заглушки Microsoft MS-DOS, підпису PE, заголовка файлу COFF (Common Object File Format) та необов'язкового заголовка. Заголовок файлу об'єкта COFF складається із заголовка файлу COFF та необов'язкового заголовка. В обох випадках за заголовками файлів негайно слідує заголовки розділів.

Заголовки PE-файлу можуть містити значно більше інформації, чим просто імпорт. Формат PE передбачає наявність заголовка з метаданими про файл, за яким іде кілька розділів, у кожному з яких знаходяться корисні відомості. На рис. 2 показано розділи, що містяться в файлі, приклад аналізу якого показаний на рис.1.

Розділ `.text` (рис. 2) містить інструкції, що виконуються центральним процесором. Згідно з описом формату файлів це єдиний розділ, що виконується, і тільки в ньому повинен зберігатися код. У всіх інших розділах перебувають дані й допоміжна інформація.

Розділ `.rdata` звичайно включає відомості про імпорт і експорт, які можна одержати за допомогою утиліт `Dependency Walker` і `Review`. У ньому також можуть зберігатися програмні дані, доступні тільки для читання. Деякі файли зберігають відомості про імпорт і експорт в розділах `.idata` і `.edata` відповідно.

Розділ `.data`. Містить глобальні дані, доступні з будь-якої частини програми. Локальні дані не можна знайти ні тут, ні в якій-небудь іншій частині PE-файлу.

Розділ `.gfids`. Функції, які є дійсними цілями непрямого виклику, перелічені в таблиці `GuardCFFunctionTable`, прикріпленій до каталогу конфігурації завантаження, що іноді називається таблицею GFIDS для стислості. Це відсортований список відносних віртуальних адрес (RVA), які містять інформацію про дійсні цілі виклику CFG. Захист потоку керування (CFG) - це високо оптимізована функція безпеки, яка створена для боротьби з уразливостями, пов'язаними з ушкодженням пам'яті. Установивши тверді обмеження на те, де додаток може виконати код, що значно утрудняє виконання експлойтами довільного коду за допомогою таких уразливостей, як переповнення буфера.

#	Name	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
1	.text	0000121D	00001000	00001400	00000400	60000020
2	.rdata	00000E48	00003000	00001000	00001800	40000040
3	.data	00000638	00004000	00000200	00002800	C0000040
4	.gfids	00000020	00005000	00000200	00002A00	40000040
5	.rsrc	000001E0	00006000	00000200	00002C00	40000040
6	.reloc	000001D8	00007000	00000200	00002E00	42000040

Рис. 2. Розділи, що містяться в файлі кейлоггера

Розділ `.rsrc` містить у собі ресурси, які використовуються файлом, що виконується, але не є його частиною, наприклад значки, зображення, меню й рядка. Рядки можуть перебувати або тут, або в головній програмі. У розділ `.rsrc` вони часто поміщаються для підтримки декількох мов.

Рис. 3. Результати аналізу ентропії

Розділ .reloc містить інформацію для переміщення бібліотечних файлів.

Дуже важлива інформація на рис. 2 міститься в колонках Virtual Size (розмір файлу в оперативній пам'яті) та Raw Size (розмір файлу на диску). Цифри в цих колонках повинні бути приблизно однаковими. Якщо значення в цих колонках занадто сильно відрізняються, то це свідчить про обфускацію, або пакування. Якщо назви розділів, що містяться в колонці name (рис. 2) відрізняються від стандартних, то це може свідчити про те, що файл упакований. При чому назва розділів буде залежати від пакувальника.

Розділи кейлоггеру, що зображені на рис.2 не містять слідів пакувальника. Це підтверджується аналізом ентропії (рис.3). Результат аналізу на рис.3 міститься в колонці Status. Аналіз якої свідчить про те, що не було знайдено ознак пакувальника.

Аналіз дизасембльованого тексту

Для отримання вичерпної інформації про програму, що аналізується необхідно її дизасемблювати та провести аналіз асемблерного тексту уривок якого наведено на рис. 4.

За адресою 0040154B (рис. 4) викликається функція SetWindowsHookExA, яка встановлює перехоплювач. Щоб його знайти проаналізуємо стек функції SetWindowsHookExA.

Параметр WH_KEYBOARD_LL, що розташований за адресою 00401549 встановлює процедуру перехоплення, яка відслідковує низькорівневі події введення з клавіатури. За адресою 00401544 міститься посилання на саму процедуру перехоплення (HookCallback). У зв'язку з тим, що аргумент, який розташований за адресою 00401540 у стеку функції SetWindowsHookExA дорівнює нулю, то процедура HookCallback пов'язана з усіма існуючими потоками, що працюють на тому робочому столі, що і викликаючий потік. Іншими словами процедура перехоплення HookCallback буде відслідковувати всі існуючі потоки на робочому столі. Цикл, що починається з адреси 00401574 не дає програмі передчасно завершитись. Основа циклу функція GetMessageA, яка вибирає сповіщення з черги програми. Завершитися цей цикл може тільки, якщо функція GetMessageA отримає сповіщення WM_QUIT. У цьому випадку функція GetMessageA поверне нульове значення, умова продовження циклу, що перевіряється за адресою 00401582, не виконається та робота програми завершиться.

Висновки

Запропоновані методи статичного аналізу строк, що містяться в файлі, що виконується, а також аналіз його дизасембльованого тексту дає змогу детально проаналізувати вміст шкідливої програми та створити ефективні мережеві та локальні сигнатури. Аналіз вмісту шкідливих програм дозволяє виявляти нові загрози та дозволяє розробляти та впроваджувати ефективні заходи для забезпечення потрібного рівня кіберзахисту.

```

.text:00401530 __stdcall WinMain(x, x, x, x) proc near
.text:00401530 msg          = tagMSG ptr -20h
.text:00401530 var_4        = dword ptr -4
.text:00401530 hInstance     = dword ptr 8
.text:00401530 hPrevInstance = dword ptr 0Ch
.text:00401530 lpCmdLine     = dword ptr 10h
.text:00401530 nCmdShow      = dword ptr 14h
.text:00401530          push    ebp
.text:00401531          mov     ebp, esp
.text:00401533          sub     esp, 20h
.text:00401536          mov     eax, __security_cookie
.text:0040153B          xor     eax, ebp
.text:0040153D          mov     [ebp+var_4], eax
.text:00401540          push    0          ; dwThreadId
.text:00401542          push    0          ; hmod
.text:00401544          push    offset HookCallback(int,uint,long) ; lpfn
.text:00401549          push    WH_KEYBOARD_LL ; idHook
.text:0040154B          call    ds:SetWindowsHookExA(x,x,x,x)
.text:00401551          mov     HHOOK_*_hook, eax
.text:00401556          test    eax, eax
.text:00401558          jnz     short loc_40156D
.text:0040156D loc_40156D:          ; CODE XREF: WinMain(x,x,x,x)+28↑j
.text:0040156D          push    esi
.text:0040156E          mov     esi, ds:GetMessageA(x,x,x,x)
.text:00401574 loc_401574:          ; CODE XREF: WinMain(x,x,x,x)+52↓j
.text:00401574          push    0          ; wParamFilterMax
.text:00401576          push    0          ; wParamFilterMin
.text:00401578          push    0          ; hWnd
.text:0040157A          lea     eax, [ebp+msg]
.text:0040157D          push    eax          ; lParam
.text:0040157E          call    esi ; GetMessageA(x,x,x,x)
.text:00401580          test    eax, eax
.text:00401582          jnz     short loc_401574
.text:00401584          mov     ecx, [ebp+var_4]
.text:00401587          xor     ecx, ebp      ; cookie
.text:00401589          pop     esi
.text:0040158A          call    __security_check_cookie(x)
.text:0040158F          mov     esp, ebp
.text:00401591          pop     ebp
.text:00401592          retn    10h
.text:00401592 __stdcall WinMain(x, x, x, x) endp

```

Рис. 4. Уривок процедури WinMain

Література

1. Svitlana Popereshnyak, Sergiy Grinenko, Olena Grinenko, Oleksandr Kovalenko, Tamara Radivilova. «*Methods for Assessing the Maturity Levels of Software Ecosystems*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 251–261.

2. Aleksandr Litvinenko, Boris Maslovsky, Oleksiy Glazok, Anton Petrov, «*Method of Optimal Planning of Cyberprotection Actions for a Corporate Information System*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 60–71.
3. Olena Kozhokhina, Olga Shcherbyna, Oleksii Chuzha, Serhii Yehorov, Maksim Iavich, Nikolay Churkin, «*Informational Technology for the Improvement of Flight Zone Security*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 262–275.
4. Oleksandr Solomentsev, Maksym Zaliskyi, Pavlo Skladannyi, «*Operation System for Modern Unmanned Aerial Vehicles*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 363–374.
5. Ivan Yashanov, Marlen Asanov, Nataliya Shakhovska, «*Method of Direct Diagnostic Problem Solution in UAV Operation System*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 390–401.
6. Marian Bedrii, Maryana Syrko, Oksana Stolyar, Oleh Kuzmin, Yurii Kryvenchuk, «*Cybercrime as a Threat to the Banking System of Ukraine and Cryptology as a Means of its Prevention*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 421–431.
7. Serhii Semenov, Denys Voloshyn, Viacheslav Davydov, «*Data Protection Method of an Unmanned Aerial Vehicle based on Obfuscation Procedure*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 515–525.
8. Viacheslav Liskin, Egor Serdobolskiy, Iryna Sopilko, Tetiana Okhrimenko, «*Two-factor User Authentication Using Biometrics*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 526–535.
9. Maksim Iavich, Arturo Arakeliani, Giorgi Iashvili, Dali Magrakvelidze, Tetiana Okhrimenko, «*Improvement of Implementation of Merkle Crypto System*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 536–545.
10. Emil Faure, Anatoly Shcherba, Yevhen Vasiliu, Andriy Fesenko, «*Cryptographic Key Exchange Method for Data Factorial Coding*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 643–664.
11. Vadym Kalchenko, Nataliia Barchenko, Andrii Tolbatov, Victor Obodiak, Volodymyr Tolbatov, Vadym Tatarinov, Sergiy Gnatyuk, «*System for Monitoring the Connection of USB Devices for Cybersecurity Auditing*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 773–785.
12. Seunghyun Yoon, Jin-Hee Cho, Dong Seong Kim, Terrence J. Moore, Frederica Free-Nelson, Hyuk Lim, «*Attack Graph-Based Moving Target Defense in Software-Defined Networks*», IEEE Transactions on Network and Service Management, 2020, Vol.: 17, Issue: 3, p. 1653 – 1668, DOI: 10.1109/TNSM.2020.2987085

References

1. Svitlana Popereshnyak, Sergiy Grinenko, Olena Grinenko, Oleksandr Kovalenko, Tamara Radivilova. «*Methods for Assessing the Maturity Levels of Software Ecosystems*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 251–261.
2. Aleksandr Litvinenko, Boris Maslovsky, Oleksiy Glazok, Anton Petrov, «*Method of Optimal Planning of Cyberprotection Actions for a Corporate Information System*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 60–71.
3. Olena Kozhokhina, Olga Shcherbyna, Oleksii Chuzha, Serhii Yehorov, Maksim Iavich, Nikolay Churkin, «*Informational Technology for the Improvement of Flight Zone Security*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 262–275.
4. Oleksandr Solomentsev, Maksym Zaliskyi, Pavlo Skladannyi, «*Operation System for Modern Unmanned Aerial Vehicles*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 363–374.

5. Ivan Yashanov, Marlen Asanov, Nataliya Shakhovska, «*Method of Direct Diagnostic Problem Solution in UAV Operation System*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 390–401.
6. Marian Bedrii, Maryana Syrko, Oksana Stolyar, Oleh Kuzmin, Yurii Kryvenchuk, «*Cybercrime as a Threat to the Banking System of Ukraine and Cryptology as a Means of its Prevention*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 421–431.
7. Serhii Semenov, Denys Voloshyn, Viacheslav Davydov, «*Data Protection Method of an Unmanned Aerial Vehicle based on Obfuscation Procedure*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 515–525.
8. Viacheslav Liskin, Egor Serdobolskiy, Iryna Sopilko, Tetiana Okhrimenko, «*Two-factor User Authentication Using Biometrics*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 526–535.
9. Maksim Iavich, Arturo Arakeliani, Giorgi Iashvili, Dali Magrakvelidze, Tetiana Okhrimenko, «*Improvement of Implementation of Merkle Crypto System*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 536–545.
10. Emil Faure, Anatoly Shcherba, Yevhen Vasiliu, Andriy Fesenko, «*Cryptographic Key Exchange Method for Data Factorial Coding*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 643–664.
11. Vadym Kalchenko, Nataliia Barchenko, Andrii Tolbatov, Victor Obodiak, Volodymyr Tolbatov, Vadym Tatarinov, Sergiy Gnatyuk, «*System for Monitoring the Connection of USB Devices for Cybersecurity Auditing*», International Workshop on Cyber Hygiene (CybHyg-2019) November 30, 2019, Kyiv, p. 773–785.
12. Seunghyun Yoon, Jin-Hee Cho, Dong Seong Kim, Terrence J. Moore, Frederica Free-Nelson, Hyuk Lim, «*Attack Graph-Based Moving Target Defense in Software-Defined Networks*», IEEE Transactions on Network and Service Management, 2020, Vol.: 17, Issue: 3, p. 1653 – 1668, DOI: 10.1109/TNSM.2020.2987085