УДК 004.491.4 **DOI:10.30837/rt.2019.3.198.14**

Р.С. ГРИНЬОВ, О.В. СЄВЄРІНОВ, канд. техн. наук

МЕТОД ПОДОЛАННЯ ЗАСОБІВ ЗАХИСТУ З ВИКОРИСТАННЯМ ВРАЗЛИВОСТЕЙ ГРАФІЧНИХ ФАЙЛІВ ФОРМАТУ BMP

Вступ

Незважаючи на використання засобів захисту, комп'ютерні віруси несуть серйозну загрозу як для простих користувачів, так і для інформаційних систем великих фірм і компаній. Сьогодні існує величезна різноманітність комп'ютерних вірусів, створюються нові методики їх приховування і поширення [1, 2]. Серед них найбільш частим є використання стеганографії для приховування вірусного програмного забезпечення в файлах [3].

На даний час для забезпечення захисту конфіденційних даних організацій використовуються антивірусне програмне забезпечення, системи виявлення вторгнень (IDS), системи запобігання вторгнень (IPS) та брандмауери [4]. Проте навіть вони не можуть гарантувати повного захисту.

Аналіз структури та особливостей файлів зображень формату BMP

Зловмисники для того, щоб приховати віруси додають до них різні нові функції та розробляють нові методики приховування шкідливого коду [5]. Для обходу антивірусних засобів, IDS/IPS і пісочниць зловмисники можуть впроваджувати вірусне програмне забезпечення в зображення.

Більшість методів аналізу файлів включають використання сигнатур вірусів і аналіз поведінки в пісочниці, а саме:

- перевірка поточного домену;
- перевірка запущених процесів;
- перевірка обсягу пам'яті;
- перевірка розміру диска;
- перевірка часу безвідмовної роботи.

Пісочниці зазвичай аналізують тільки виконувані файли, бібліотеки динамічних посилань (DLL), документи, що були створені в пакеті Microsoft Office та подібних, аплети Java [6]. Більшість із засобів захисту просто не реагують на файли зображень або інший безпечний тип файлів, оскільки вважається, що немає причин витрачати процесорний цикл на аналіз зображення [7].

Проаналізуємо можливість використання файлу зображення в якості контейнеру для непомітного транспортування комп’ютерного вірусу, ускладнення розбору інциденту інформаційної безпеки, приховування факту проникнення в систему, а також канали та методики, що при цьому використовуються.

Розглянемо формат файлів зображень BMP. Кожен файл цього формату має заголовок файлу, заголовок зображення, растрові данні та карту кольорів (крім зображень з 24-бітним кольором). Структура заголовку файлу формату BMP представлена в табл. 1. Він являє собою 14-байтну структуру, що знаходиться на початку файлу та містить в собі інформацію про тип та розмір файлу та розташування даних. Далі в файлі формату BMP знаходиться заголовок зображення, що містить інформацію про розмір, колір та стиск зображення (див. табл. 2).

В полі Compression визначається тип стиску. Якщо значення поля дорівнює 0, то стиск відсутній. Якщо значення RLE-4 або RLE-8, то використовується метод стиску груповими координатами відповідно із 4-біт/піксель та 8-біт/піксель.

Таблиця 1

Структура заголовку файлу формату BMP

Зміщення	Розмір (байт)	Ім’я	Опис
0	2	Type	Сигнатура формату. Використовується для ідентифікації формату. Має бути 4D42(hex)/424D(hex) (little-endian/big-endian). Після приведення до системи ASCII-символів має вигляд “BM”.
2	4	Size	Розмір файлу в байтах.
6	2	Reserved 1	Зарезервоване поле має містити 0.
8	2	Reserved 2	Зарезервоване поле має містити 0.
10	4	OffsetBits	Положення піксельних даних відносно початку файлу (в байтах).

Таблиця 2

Структура заголовку зображення

Зміщення	Розмір (байт)	Ім’я	Опис
14	4	Size	Довжина заголовку.
18	4	Width	Ширина зображення в пікселях.
22	4	Height	Висота зображення в пікселях.
26	2	Planes	Кількість площин.
28	2	BitCount	Глибина кольору, біт на піксель (1, 4, 8, 24).
30	4	Compression	Тип компресії (0 – відсутня, 1 – RLE-8, 2 – RLE-4).
34	4	SizeImage	Розмір зображення, байт (включно з доповненням).
38	4	XpelsPerMeter	Горизонтальна роздільна здатність, пікселів на метр.
42	4	YpelsPerMeter	Вертикальна роздільна здатність, пікселів на метр.
46	4	ColorsUsed	Число кольорів, що використовується (0 – максимально можливе для даної глибини кольору).
50	4	ColorTable	Кількість основних кольорів.

Найбільшу увагу привертають поля Size (розмір файлу BMP в байтах), XpelsPerMeter та YpelsPerMeter (горизонтальна та вертикальна роздільна здатність, пікселів на метр) та зарезервовані поля, адже вони є ненадійними [8]. Звичайний заголовок BMP починається з подібного рядка 42 4D XX XX XX XX 00 00 00 00.

Так наприклад, зображення з візерунком шахової дошки (рис. 1) має заголовок представлений на рис. 2.

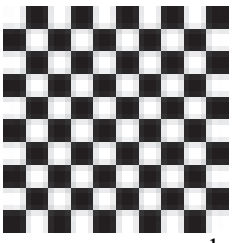


Рис. 1. Приклад малюнку в форматі BMP

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Текст декодиров
00000000   42 4D 4A 13 00 00 00 00 00 00 8A 00 00 00 7C 00  BMJ.......Љ...|
00000010   00 00 28 00 00 00 28 00 00 00 01 00 18 00 00 00  ..(...(.........
00000020   00 00 C0 12 00 00 C3 0E 00 00 C3 0E 00 00 00 00  ..А...Г...Г.....
00000030   00 00 00 00 00 00 00 00 FF 00 00 FF 00 00 FF 00  ........я..я..я
00000040   00 00 00 00 00 FF 42 47 52 73 80 C2 F5 28 60 B8  .....яBGRsЂВх(`
```

Рис. 2. Заголовок файлу зображення

На рис. 3 представлені значення полів початку заголовку файлу з попереднього прикладу.

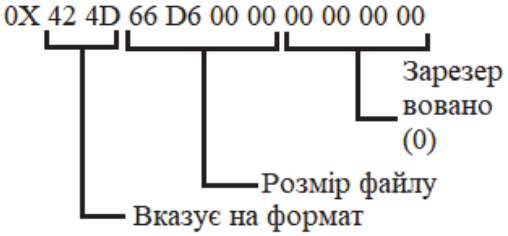


Рис. 3. Значення полів заголовку зображення

Існує можливість зміни зарезервованих полів та полів, що містять інформацію про розмір файлу в будь-якому hex-редакторі. Запишемо наприклад в ці ділянки слово “testtest” (рис. 4).

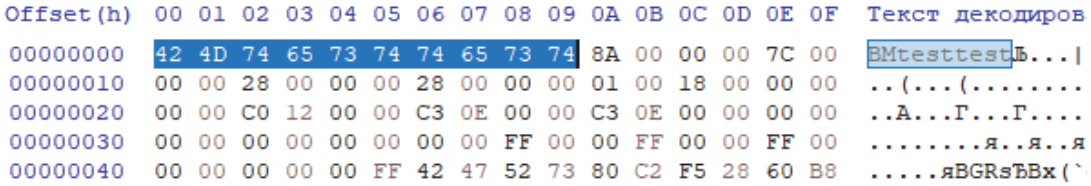


Рис. 4. Зображення зі зміненим заголовком

Тепер заголовок зображення матиме вигляд (рис. 5).

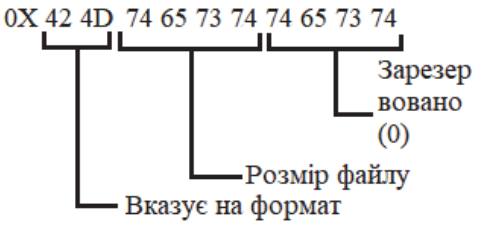


Рис. 5. Змінений заголовок зображення

Проте на самому зображенні не буде ніяких спотворень (рис. 6), оскільки поля службові і не впливають на відображення інформації. Крім того розмір файлу також не змінюється.

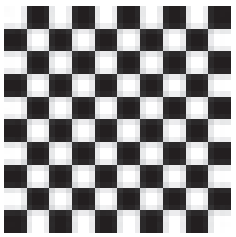


Рис. 6. Зображення зі зміненими полями

Як вже згадувалося раніше заголовок файлу BMP починається з 0x 42 4D, що вказує на тип файлу (BM). При конвертації в інструкції асемблера отримаємо, що 42 – це inc edx, а 4D – dec ebp. Це означає, що якщо ці інструкції будуть передувати коду програми, то вони не викличуть збоїв та не мають команд переходів.

В hex-редакторі змінимо значення декількох останніх рядків (рис. 7).

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Текст декодиров
00001240	13	13	E9	E9	E9	FF	FF	FF	FF	FF	FF	EC	EC	EC	16	16	..ййяяяяяяммм.
00001250	16	00	00	00	01	01	01	00	00	00	FC	FC	FC	FD	FD	FD	..........ьььээ
00001260	FF	FF	FF	EA	EA	EA	17	54	68	69	73	20	70	6C	61	63	яяяккк.This pla
00001270	65	20	69	73	20	65	6E	6F	75	67	68	20	74	6F	20	61	e is enough to
00001280	63	63	6F	6D	6D	6F	64	61	74	65	20	6D	61	6C	69	63	ccommodate mali
00001290	69	6F	75	73	20	63	6F	64	65	2C	20	77	68	69	63	68	ious code, whic
000012A0	20	77	69	6C	6C	20	62	65	20	68	69	64	64	65	6E	20	will be hidden
000012B0	69	6E	20	74	68	65	20	69	6D	61	67	65	2E	20	42	75	in the image. B
000012C0	74	20	64	75	65	20	74	6F	20	74	68	65	20	63	68	61	t due to the ch
000012D0	6E	67	65	73	20	74	68	65	72	65	20	77	69	6C	6C	20	nges there will
000012E0	62	65	20	6E	6F	74	69	63	65	61	62	6C	65	20	76	69	be noticeable v
000012F0	73	75	61	6C	20	64	69	73	74	6F	72	74	69	6F	6E	73	sual distortion
00001300	2E	20	49	66	20	61	72	74	69	66	69	63	69	61	6C	6C	. If artificial
00001310	79	20	72	65	64	75	63	65	20	74	68	65	20	68	65	69	y reduce the he
00001320	67	68	74	20	6F	66	20	74	68	65	20	69	6D	61	67	65	ght of the imag
00001330	2C	20	74	68	65	6E	20	74	68	65	79	20	63	61	6E	20	, then they can
00001340	62	65	20	68	69	64	64	65	6E	2E							be hidden.

Рис. 7. Змінене зображення

Відкривши отримане зображення ми побачимо, що у правому верхньому куті з’явилися спотворені пікселі (рис. 8).

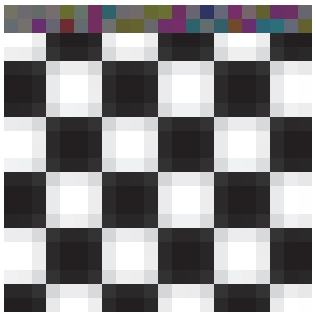


Рис. 8. Збільшений правий кут зміненого зображення

Скориставшись hex-редактором можна змінити висоту зображення і приховати спотворені пікселі (рис. 9).

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Текст декодиров
00000000	42	4D	74	65	73	74	74	65	73	74	8A	00	00	00	7C	00	BMtesttestЉ...|
00000010	00	00	28	00	00	00	26	00	00	00	01	00	18	00	00	00	..(...&.....
00000020	00	00	C0	12	00	00	C3	0E	00	00	C3	0E	00	00	00	00	..А...Г...Г....
00000030	00	00	00	00	00	00	00	00	FF	00	00	FF	00	00	FF	00	........я..я..я
00000040	00	00	00	00	00	FF	42	47	52	73	80	C2	F5	28	60	B8	.....яBGRsЂВх(`

Рис. 9. Штучне зменшення висоти зображення на 1 піксель

Після здійснення цих операцій при перегляді зображення спотворені пікселі не будуть відображатися на екрані (рис. 10).

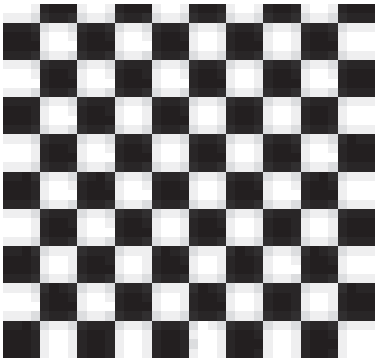


Рис. 10. Зображення зі штучно зменшеною висотою

Таким чином, використовуючи зображення формату BMP можна провадити в нього вірус, штучно зменшивши висоту зображення, який не буде помітним для користувача (на зображенні не буде спотворених пікселів). Ін'єкція можлива через те, що байти, які вказують на тип файлу, з яких і починається файл, BM в ASCII, в шістнадцятковому вигляді – 42 4D, при конвертації в інструкції асемблера не призводять до помилки виконання, а подальші 8 байт заголовка ніяк не впливають на інтерпретацію зображення. Ці 8 байт можна заповнити будь-якими інструкціями асемблера, наприклад, записати в них jmp-інструкцію, яка вкаже на вірус, що зберігається в зображенні [8]. Наприклад, в нашому випадку функція jmp могла вказувати на адресу 0000D583, де починається текст.

За допомогою нескладних дій можливо отримати зображення, що містить вірус. Така можливість обумовлена особливостями формату зображення BMP. Такий підхід приховування вірусу дозволить уникнути аналізу та виявлення шкідливого коду пісочницями. Проте IPS/IDS та антивіруси можуть виконати статичний аналіз та виявити вірус [8]. Для приховування шкідливого коду від цих засобів захисту необхідно використати обфускацію (заплутування коду) [7]. Найпростішим варіантом є використання найпростіших логічних операцій для кодування, таких як XOR. В кості ключа можна використати 32-х бітне значення в діапазоні від 0x11111111 до 0xffffffff. Обфускація буде досягатися за рахунок сили ключа, це дозволить уникнути жорстко закодованого коду.

Наприклад, припустимо, що ключ дорівнює 0x39643964. В цьому випадку маємо результат представлений на рис. 11.

$$
\begin{array}{ccccc}
0x54455445 & + & 0xCCCCCCCC & + & 0xCCCCCCCC \\
\oplus & & \oplus & & \oplus \\
0x39643964 & + & 0x39643964 & + & 0x39643964 \\
= & & = & & = \\
0x6d216d21 & + & 0xf5a8f5a8 & + & 0xf5a8f5a8
\end{array}
$$

Рис. 11. Приклад результату операції обфускації

Щоб виконати код, що зберігається в зображенні, можна використати корисне навантаження PowerShell, воно завантажить зображення та виконає код в пам’яті. Ми використаємо готове рішення від Cobalt Strike. Воно використовує System.Net.WebClient для завантаження файлу, в нашому випадку це зображення, а потім VirtualAlloc та CreateThread для зчитування та виконання вірусу.

Проведений аналіз підтверджує можливість використання зображень формату BMP з впровадженим вірусним кодом для подолання засобів захисту. Існує багато варіантів завантаження зображення з впровадженим вірусом та виконання вірусного коду. Це можна зробити за допомогою будь-якого виконуваного файлу або скрипта, впровадити код у вже існуючий виконуваний файл. Такі методи дозволять оминути засоби захисту. Адже під час сканування засоби захисту не виявлять у них вірус. Проте більш перспективним є використання апаратних закладок у поєднанні з цим методом [9, 10].

Аналіз виявлення вірусних зразків різними антивірусними засобами

В ході дослідження була розроблена програма, яка отримує в якості вхідних даних необроблене корисне навантаження, що може бути згенеровано за допомогою Metasploit та зображення формату BMP. Тестування проводилось на персональному комп’ютері під керуванням операційної системи Parrot Security OS. Metasploit – це інструмент, призначений для створення та використання експлойтів, а також експлуатації та постексплуатації вразливостей. Для створення вхідних даних також можна скористатися утилітою Msfvenom, яка є складовою Metasploit [11]. Розроблена програма впроваджує в зображення вірус, змінює ви-

соту зображення та записує в заголовок jump-функцію. Після чого запускає web-сервер на комп'ютері з інфікованим зображенням та створює команду PowerShell, яка завантажую зображення та виконує вірус.

Для перевірки файлів використовувались ресурси web-сайту www.virustotal.com. Він надає можливість аналізу файлу по базам даних всіх популярних антивірусних засобів та безпосереднього аналізу файлу антивірусами. Результат сканування шелл-коду в форматі raw, що був згенерований за допомогою msfvenom представлені на рис. 12.

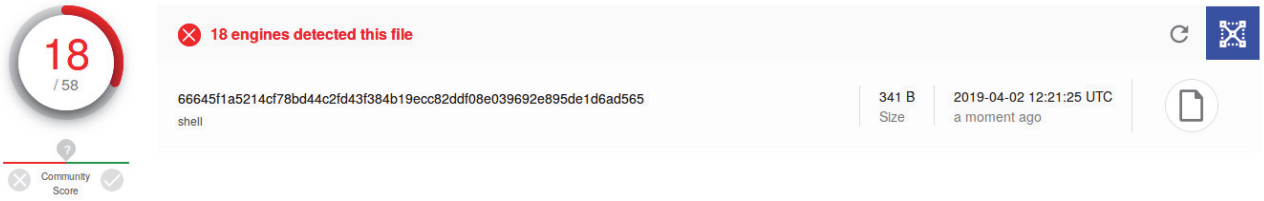


Рис. 12. Результат сканування шелл-коду в форматі raw

18 з 58 антивірусних засобів визначили тестовий файл, як вірусний. Той самий вірусний код але у вигляді виконуваного файлу визначили вірусним вже 53 з 70 антивірусів (рис. 13). Різна кількість антивірусних засобів, що перевіряла файл зумовлена типом файлу.

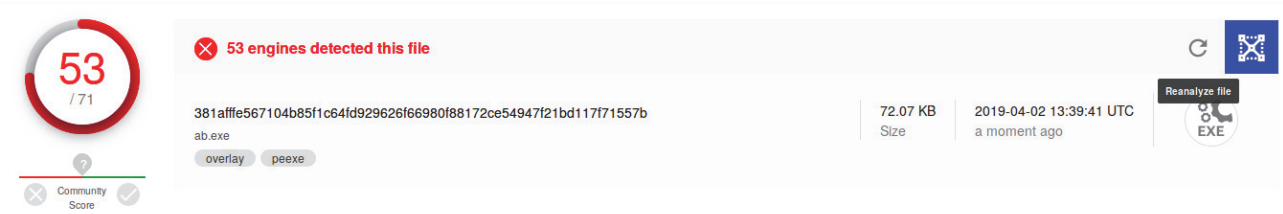


Рис. 13. Результати сканування шелл-коду в форматі exe

В дослідженнях була перевірена можливість виявлення вірусу, що був прихований в програму PuTTY – безкоштовний клієнт для протоколів Telnet, SSH. Результати перевірки оригінального файлу PuTTY представлені на рис. 14.

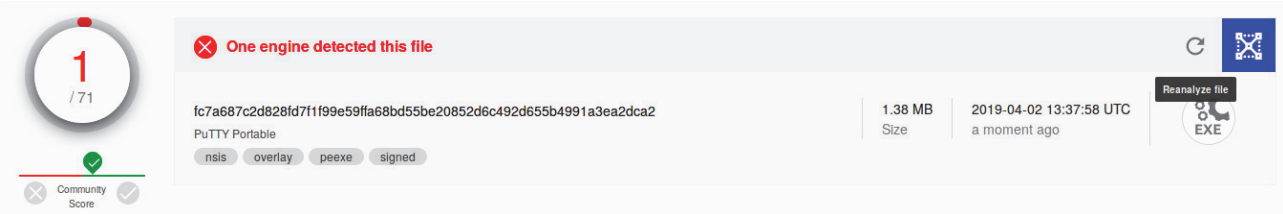


Рис. 14. Результати сканування оригінального PuTTY

Як видно з результатів, лише один антивірус зробив припущення про наявність вірусу (рис. 15).

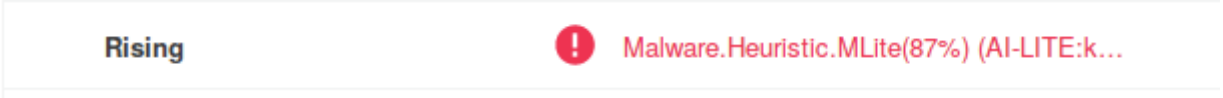


Рис. 15. Припущення про наявність вірусу

Для впровадження вірусу у виконуваний файл були використані дві найпоширеніші методики, що дозволяють зберегти працездатність файлу: створення нової секції та впровадження у code cave. Code cave – це послідовність нульових байтів в пам'яті процесу. Для впровадження вірусу була використана утиліта Backdoor Factory – спеціальна утиліта для впровадження шелл-коду у виконувані файли та динамічні бібліотеки. Спочатку був видале-

ний цифровий підпис файлу. Результати перевірки PuTTY з видаленим цифровим підписом представлені на рис. 16.

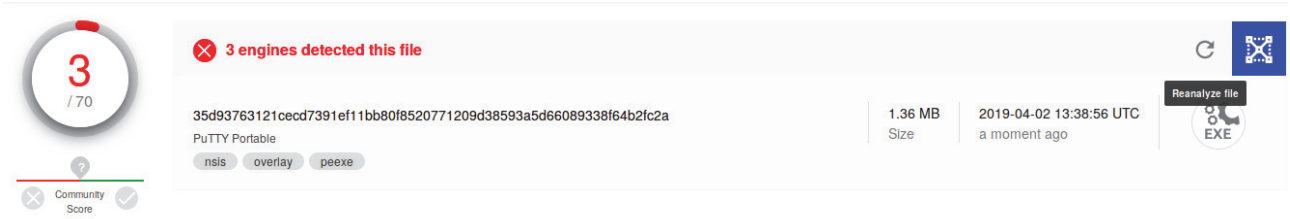


Рис. 16. Перевірка PuTTY з видаленим цифровим підписом

На рис. 17 представлені результати сканування файлу PuTTY з новою порожньою секцією (без вірусу) в оригінальному файлі.

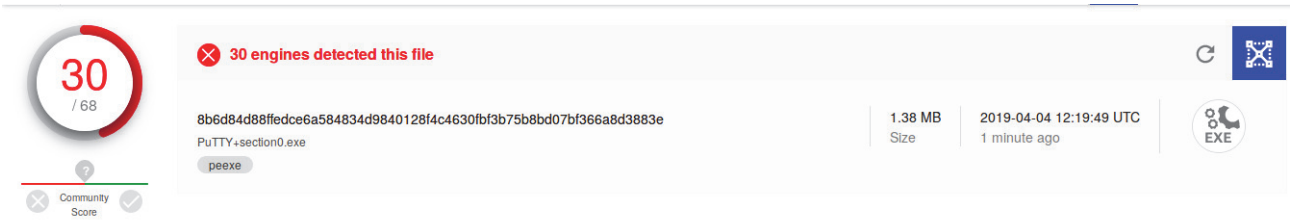


Рис. 17. Перевірка PuTTY з новою порожньою секцією

Як видно з результатів після створення нової порожньої секції без впровадження вірусного коду файл визначається 30 антивірусними засобами як вірус. Після створення нової секції та впровадження в неї вірусу 32 з 70 антивірусних засобів його виявили (рис. 18).

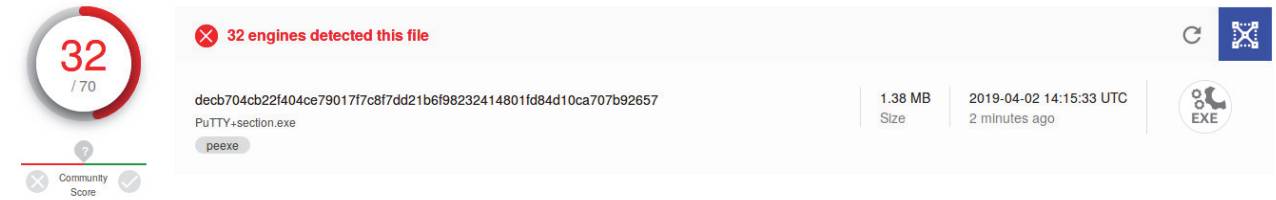


Рис. 18. Результати приховування вірусу в новій секції

В дослідженнях була перевірена можливість виявлення вірусу, що був прихований у code cave (рис. 19).

```
└─$ backdoor-factory -f PuTTY.exe -s user_supplied_shellcode_threaded -U section -o PuTTY+cave.exe -Z
```

Рис. 19. Впровадження вірусу в code cave

За результатами сканування лише 9 антивірусів вивили вірус (рис. 20).

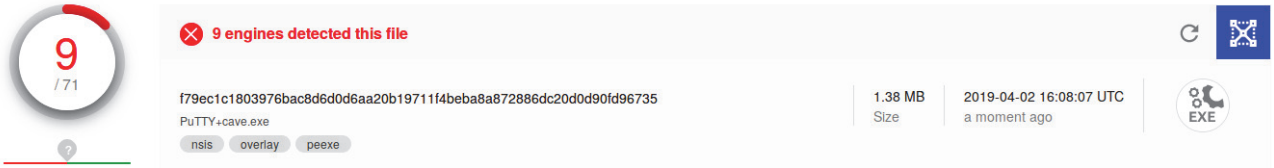


Рис. 20. Результати приховування вірусу в code cave

В дослідженнях був перевірений оригінальний файл BMP (рис. 21). Після цього був перевірений файл, в який був впроваджений шелл-код за допомогою розробленої програми (рис. 22).

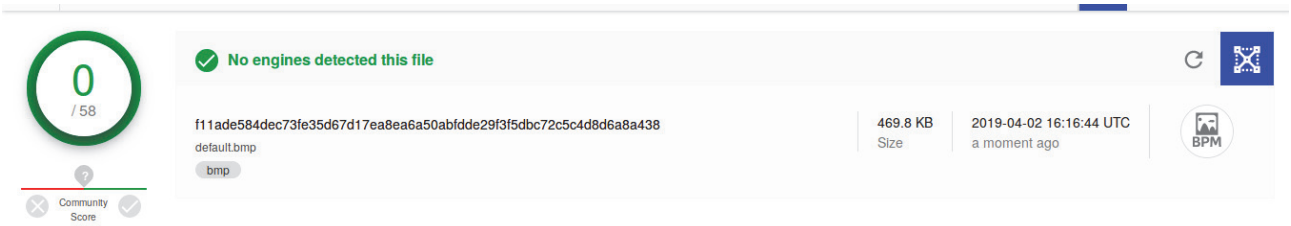


Рис. 21. Результати перевірки оригінального зображення BMP

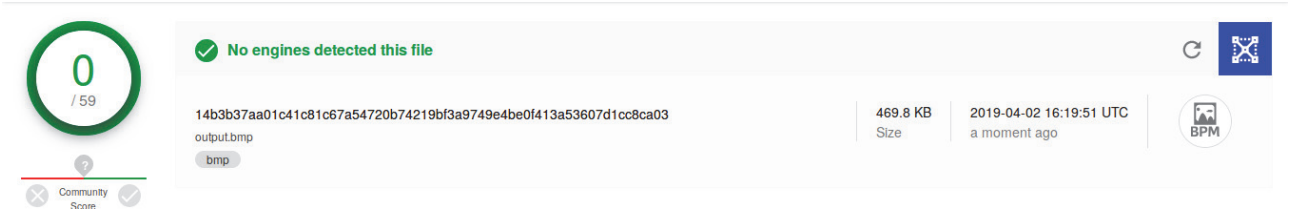
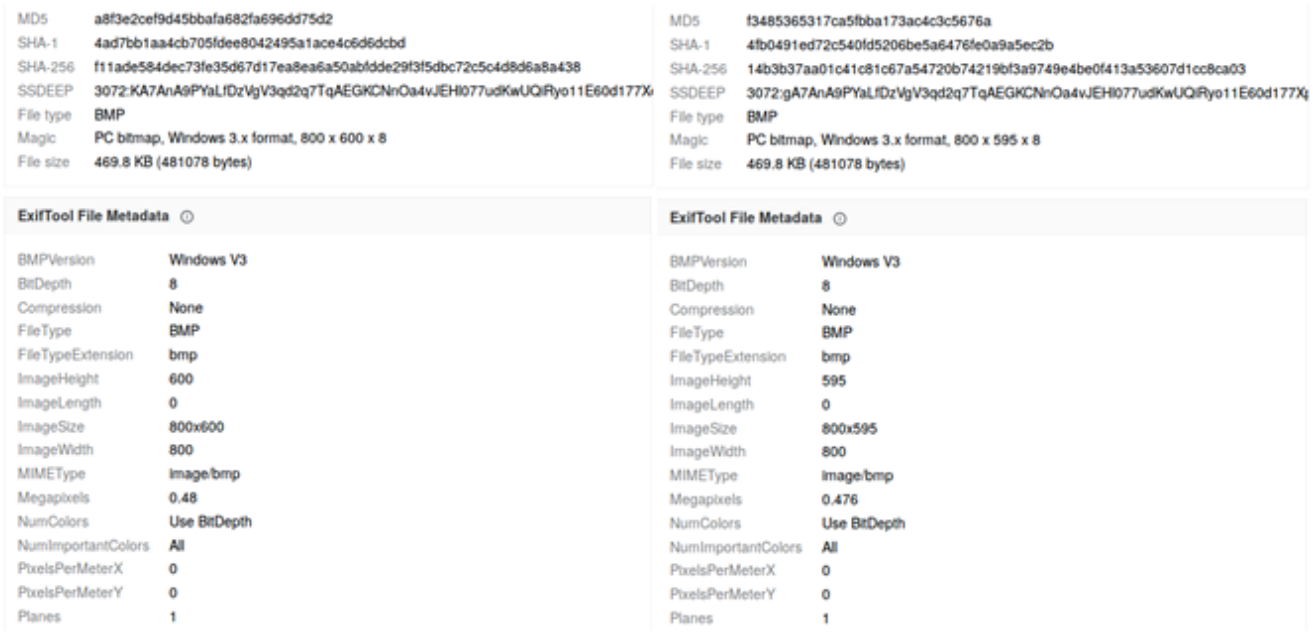


Рис. 22. Результати перевірки інфікованого зображення BMP

Як видно з результатів перевірки жоден антивірусний засіб не зміг знайти вірус у зображенні. Що означає, що метод приховування вірусів від засобів захисту, в тому числі й антивірусів, розроблений в цій роботі є найбільш ефективним.

Дослідження показали, що детальна інформація про оригінальне (рис. 23, *а*) та інфіковане зображення (рис. 23, *б*), відрізняється лише значеннями висоти в пікселях та значеннями хеш функцій, вирахуваних від файлів.



а *б*

Рис. 23. Детальна інформація про зображення

Узагальнені результати проведених перевірок наведено у табл. 3. Знаком “-” позначається, якщо антивірус не виявив вірус, знаком “+” , якщо вірус був виявлений, знаком “*”, якщо антивірус не обробляє цей тип файлу.

Таблиця 3

Результати сканування зразків

Назва антивірусу	Шелл-код у фо-рматі raw	Шелл-код у фо-рматі exe	PuTTY	PuTTY без циф-рового підпису	PuTTY з порож-ньою новою секцією	PuTTY з шелл-кодом в новій секції	PuTTY з шелл-кодом в code cave	Оригінальне BMP зображення	Інфіковане BMP зобра-ження
Acronis	*	+	-	-	-	-	-	*	*
Ad-Aware	+	+	-	-	-	-	-	-	-
AegisLab	+	+	-	-	+	+	+	-	-
AhnLab-V3	+	+	-	-	-	-	-	-	-
Alibaba	*	+	-	-	-	-	-	*	*
ALYac	+	+	-	-	-	-	-	-	-
Antiy-AVL	-	-	-	-	-	-	-	-	-
Arcabit	+	+	-	-	-	-	-	-	-
Avast	+	+	-	-	+	+	-	-	-
Avast Mobile Security	-	-	-	-	-	-	-	-	-
AVG	+	+	-	-	+	+	-	-	-
Avira	-	+	-	-	+	+	-	-	-
Babable	-	-	-	-	-	-	-	-	-
Baidu	-	-	-	-	-	-	-	-	-
BitDefender	+	+	-	-	-	-	-	-	-
Bkav	-	+	-	+	+	+	+	-	-
CAT-QuickHeal	-	+	-	-	-	-	-	-	-
ClamAV	+	+	-	-	+	+	-	-	-
CMC	-	-	-	-	-	-	-	-	-
Comodo	-	+	-	-	+	+	-	-	-
CrowdStrike Falcon	*	+	-	-	+	+	+	*	*
Cybereason	*	+	-	-	-	-	-	*	*
Cylance	*	+	-	-	+	+	+	*	*
Cyren	-	+	-	-	+	+	-	-	-
DrWeb	+	+	-	-	+	+	-	-	-
eGambit	*	+	-	-	+	+	+	*	*
Emsisoft	+	+	-	-	-	-	-	-	-
Endgame	*	+	-	-	+	+	-	*	*
eScan	+	+	-	-	-	-	-	-	-
ESET-NOD32	-	+	-	-	+	+	-	-	-
F-Prot	-	+	-	-	-	+	-	-	-
F-Secure	-	+	-	-	+	+	-	-	-
FireEye	+	+	-	-	+	+	-	-	-
Fortinet	-	+	-	-	+	+	-	-	-
GData	+	+	-	-	-	-	-	-	-
Ikarus	-	+	-	-	-	+	+	-	-
Jiangmin	-	-	-	-	-	-	-	-	-
K7AntiVirus	-	+	-	-	-	-	-	-	-
K7GW	-	+	-	-	-	-	-	-	-
Kaspersky	+	+	-	-	+	+	-	-	-
Kingsoft	-	-	-	-	-	-	-	-	-
Malwarebytes	-	-	-	-	-	-	-	-	-
MAX	+	+	-	-	-	-	-	-	-
MaxSecure	*	-	-	-	-	-	-	*	*
McAfee	-	+	-	-	+	+	-	-	-
McAfee-GW-Edition	-	+	-	+	+	+	-	-	-
Microsoft	+	+	-	-	+	+	+	-	-

Продовження табл. 3

Назва антивірусу	Шелл-код у фор-маті raw	Шелл-код у фо-рматі exe	PuTTY	PuTTY без циф-рового підпису	PuTTY з порожньою новою секцією	PuTTY з шелл-кодом в новій секції	PuTTY з шелл-кодом в code cave	Оригінальне BMP зображення	Інфіковане BMP зобра-ження
NANO-Antivirus	+	+	-	-	+	+	-	-	-
Palo Alto Networks	*	-	-	-	-	-	-	*	*
Panda	-	-	-	-	-	-	-	-	-
Symantec	+	-	-	-	-	-	-	*	*
Symantec Mobile Insight	*	-	-	-	-	-	-	*	*
Qihoo-360	-	+	-	-	+	+	-	-	-
Rising	-	+	+	+	+	+	+	-	-
SentinelOne	*	+	-	-	+	+	-	*	*
Sophos AV	-	+	-	-	+	+	-	-	-
Sophos ML	*	+	-	-	-	-	-	*	*
SUPERAntiSpyware	-	+	-	-	-	-	-	-	-
Symantec	*	+	-	-	+	+	+	-	-
TACHYON	-	-	-	-	-	-	-	-	-
Tencent	+	-	-	-	-	-	-	-	-
TheHacker	-	-	-	-	-	-	-	-	-
TotalDefense	-	-	-	-	-	-	-	*	*
Trapmine	*	+	-	-	-	-	-	*	*
TrendMicro	-	+	-	-	-	-	-	-	-
TrendMicro-HouseCall	-	+	-	-	-	-	-	-	-
Trustlook	*	-	-	-	-	-	-	*	*
VBA32	-	-	-	-	-	-	-	-	-
VIPRE	-	-	-	-	+	-	-	-	-
ViRobot	-	+	-	-	-	-	-	-	-
Webroot	*	+	-	-	-	-	-	*	*
Yandex	-	+	-	-	+	+	-	-	-
Zillya	-	-	-	-	-	+	-	-	-
ZoneAlarm	+	+	-	-	+	+	-	-	-
Zoner	-	-	-	-	-	-	-	-	-

Висновки

У ході досліджень була розроблена та продемонстрована концепція атаки, що дозволяє приховати вірус у зображеннях BMP. Така можливість зумовлена особливостями структури даного формату.

Проаналізувавши отримані результати, можна стверджувати, що даний метод приховування вірусу дозволяє подолати всі засоби захисту та приховати шкідливий код. Крім того подібний підхід може суттєво ускладнити розбір інциденту інформаційної безпеки. Це обумовлено тим, що більшість із засобів захисту просто не звертають уваги на зображення або інший безпечний тип файлу. Оскільки вважають, що немає причин витрачати процесорний цикл на аналіз зображення. Антивірусні засоби захисту під час сканування не виявили вірус у файлі формату BMP. Крім того аналіз поведінки зображення в "пісочниці" також не виявить вірус, оскільки при відкритті зображення шкідливий код не почне виконуватися. Враховуючи вище сказане IDS, IPS також не зможуть виявити вірус. Це свідчить про неефективність виявлення та протидії існуючих засобів захисту розглянутій атаці.

За результатами випробувань можна стверджувати, що розроблений метод приховування вірусного коду в зображенні формату BMP є найбільш ефективним, оскільки шелл-код не був виявлений жодним засобом захисту. Результати даної роботи можна використовувати під час розробки засобів антивірусного захисту та комплексних засобів захисту ІТС та для їх модернізації з метою попередження подібних атак.

При цьому необхідно враховувати, що навіть звичайні файли з простою структурою без підтримки скриптів можуть становити серйозну загрозу. Тому необхідно розроблювати нові більш ефективні засоби захисту.

Основна небезпека подібних зображень з вірусами полягає в тому, що для виявлення загрози необхідно використовувати нестандартні методи. Можна змінити налаштування засобів захисту, щоб вони перевіряли всі типи файлів, але це суттєво сповільнить або навіть повністю паралізує роботу всієї інформаційно-комунікаційної системи.

Список літератури:

1. Гриньов Р.С. Аналіз тенденцій вірусних загроз в Україні / Гриньов Р.С., Сєвєрінов О.В. // Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління: міжнар. конф. Харків, 2019. С. 100.

2. Гриньов Р.С. Аналіз статистики та особливостей розповсюдження вірусів в Україні // Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління: міжнар. конф. Харків, 2019. С. 100.

3. Pare. Virus spread over networks: Modeling, analysis, and control : Ph.D. Electrical & Computer Eng / Pare. University of Illinois at Urbana-Champaign, 2018.

4. Jingwei LEI. Virus program detection method, terminal, and computer readable storage medium. United States, 2018. 19 c.

5. Wen-Kwang Tsao. Detecting malicious code in sections of computer files / Wen-Kwang Tsao, Pinghuan Wu, Zipan Bai. United States, 2018. 15 c.

6. Lubomir Sikora. Swarm Virus, Evolution, Behavior and Networking / Lubomir Sikora, Ivan Zelinka. Berlin, 2017.

7. Carey Parker. Computer Security. North CarolinaUSA, 2018.

8. Гриньов Р.С. Аналіз небезпеки впровадження вірусного програмного забезпечення в зображення / Гриньов Р.С., Севєрінов О.В. // Комп'ютерні та інформаційні системи і технології: міжнар. наук.-техн. конф. Харків, 2019. С. 75.

9. Гриньов Р.С. Шкідливий USB HID-емулятор / Гриньов Р.С., Сєвєрінов О.В. // Радіоелектроніка та молодь у XXI столітті: міжнар. форум. Харків, 2018. С. 120-121.

10. Гриньов Р.С. Аналіз небезпеки апаратних закладних пристроїв / Гриньов Р.С., Сєвєрінов О.В. // Радіоелектроніка та молодь у XXI столітті: міжнар. форум. Харків, 2019. С. 93-94.

11. Офіційна документація Parrot Security OS. URL: https://docs.parrotsec.org/doku.php (дата звернення: 22.10.2018).

Харківський національний
університет радіоелектроніки

Надійшла до редколегії 04.09.2019