Міністерство освіти і науки України
Хмельницький національний університет

АПКН 2019

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XI всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2019»

14-15 листопада 2019

Том 1
Роботи студентів та молодих вчених Факультету програмування та комп'ютерних і телекомунікаційних систем ХНУ

Хмельницький 2019

Актуальні проблеми комп'ютерних наук. Збірник наукових праць за матеріалами XI всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» – Хмельницький: ХНУ, 2019, Т.1. – 248с.

У збірнику наукових праць представлені перспективні практичні розробки аспірантів, магістрів та здобувачів в області сучасних інформаційних технологій. Розглянуто актуальні проблеми комп'ютерних наук, прикладної математики й інформатики, приведено ряд робіт по впровадженню інформаційних технологій у виробництво та управління. Висвітлено перспективні розробки з сучасних систем пошуку й обробки інформації, САПР та математичного, комп'ютерного і нейромережевого моделювання.

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК - 2019

XI всеукраїнська науково-практична конференція

Метою конференції є висвітлення актуальних проблем комп'ютерних наук, інформатики та інформаційних технологій.

Основні напрямки роботи конференції:

1. Прикладні інформаційні технології.
2. Сучасні системи пошуку, захисту і обробки інформації.
3. Математичне, комп'ютерне і нейромережеве моделювання.
4. САПР, математичні моделі і методи рішення інженерних задач.
5. Проблеми впровадження інформаційних технологій у виробництво та управління.

Робочі мови конференції: українська, англійська.

КЕРІВНИЦТВО ОРГКОМІТЕТУ:

СИНЮК О. М.
голова оргкомітету, проректор Хмельницького національного університету з наукової роботи, доктор технічних наук, професор

СОРОКАТИЙ Р. В.
заступник голови оргкомітету, доктор технічних наук, завідувач кафедри Комп'ютерних наук та інформаційних технологій ХНУ, професор

БАРМАК О. В.
заступник голови оргкомітету, доктор технічних наук, професор

СЕКРЕТАРІАТ КОНФЕРЕНЦІЇ:

Мазурець О. В.
секретар конференції, старший викладач каф. КНІТ ХНУ

КОНТАКТНА ІНФОРМАЦІЯ:
e-mail для листування: *apkt.khnu@gmail.com*

ЗМІСТ

УДК 004.4

Бородін М.Ю, Манзюк Е.А, Скрипник Т.К.

Хмельницький національний університет

ЗАБЕЗПЕЧЕННЯ ЗАХИЧЕНОСТІ ПРОГРАМНИХ СИСТЕМ З ВИКОРИСТАННЯМ ТРАНСФОРМАЦІЙНИХ ПЕРЕТВОРЮВАНЬ ВИКОНУЮЧОГО КОДУ

Розглянуто інформаційну технологію з захисту програмних продуктів, яка дозволяє перетворювати скомпільований код таким чином що його робота буде можливою лише з допомогою інтерпретатора. Перетворений код має вигляд випадкового набору даних та не підлягає розумінню без попереднього аналізу з допомогою покрокового трасування коду інтерпретатора. Наведено архітектуру системи захисту програмного забезпечення з використанням перетворювання коду.

Software protection information technology, which allows you to transform compiled code in such a way that its work will be possible only with the help of an interpreter, is considered. The converted code looks like a random data set and is not understandable without step by step tracing of the interpreter code. The architecture of the software protection system using code conversion is given.

З кожним роком системи захисту програмного забезпечення стають досконалішими та доступнішими. Проте й засоби для роботи з таким захистом теж вдосконалюються. На даний момент будь-який захист з умовною застарілістю рік вважається «публічним». Це означає що з високою долею вірогідності покрокові описи роботи з цим захистом можуть бути у вільному доступі, вже не кажучи про команди ентузіастів чи професійні компанії які спеціалізуються на цьому.

Існує безліч програмних засобів які дозволяють тим чи іншим чином попередити несанкціоноване втручання в цілісність програмного забезпечення. Серед них можна виділити два основні типи: пакери та обфускатори.

Робота пакерів полягає в стисненні виконуваного файлу і прикріплення до нього коду, необхідного для розпакування і виконання вмісту файлу. Упаковка застосовується по ряду причин:

– Упакований файл займає менше місця на носії інформації, що допомагає прискорити його завантаження в пам'ять;

– Деякі види упаковки суміщені з шифруванням вмісту файлу для запобігання зворотного розробки програми;

– Упаковка з шифруванням може використовуватися і зловмисно при створенні вірусів, щоб зашифрувати і видозмінити код вірусу для утруднення виявлення антивірусами.

З цього випливає що принцип роботи пакерів схожий архіватором. Забезпечує мінімальний рівень захисту оскільки в кінцевому випадку сам розпаковує та запускає потрібний модуль. Використання самого лиш пакера для програм, на платформі .Net, ставить під загрозу стійкість програми, оскільки його повна розпаковка вимагає лише перехоплення завантаження модулю чи дампу пам'яті. Серед найвідоміших пакерів можна виділити: ASPack, eXPressor, Mpress, PECompact, PELock, UPX, PESpin, Confuser, ConfuserEx.

Обфускація або заплутування коду [1] – приведення початкового коду або виконуваного програмного коду до вигляду, який зберігає його функціональність, але ускладнює аналіз, розуміння алгоритму роботи і модифікації при декомпіляції. Принцип роботи обфускаторів дещо складніший ніж у пакерів оскільки за мету взято максимально заплутати основний код а не приховати його шаром пакеру. Максимальна ефективність роботи досягається з допомогою поєднання кількох обфускаторів з пакером(пакерами). Серед них можна виділити DotFuscator, Agile.Net, Confuser, ConfuserEx, Themida, The Enigma Protector, VMProtector.

Одним із видів обфускації на базі перетворення коду є віртуалізація [2]. Цей принцип захисту поєднує в собі два попередніх принципи та забезпечує максимальну стійкість до несанкціонованого втручання. Як і у випадку обфускації - код поділяється на окремі команди. Але кожна машинна команда змінюється не на набір аналогічних машинних, а на псевдокод, що описує її для інтерпретатора захисту. Потім посилання на оброблений код передається в функцію виконання, яка інтерпретує псевдокод і виконує його командами процесора. Серед програмних засобів які реалізують подібну технологію можна виділити EazFuscator, The Enigma Protector, VMProtector, Koi.VM.

Системии захисту програмного забезпечення на базі перетворювання коду називають віртуалізацєю. Віртуалізація коду вважається найсучаснішою технологією ускладнення та заплутування коду, що значно збільшує стійкість програми до аналізу і взлому. Як і у випадку обфускації – код розділяється на окремі команди. Але кожна машинна команда змінюється не на набір аналогічних машинних, а на псевдокод, який описує її для інтерпретатора захисту. Після чого посилання на оброблений код передається в функцію виконання, яка інтерпретує псевдокод і виконує його командами процесора.

Складність аналізу такого коду полягає в тому, що псевдокод унікальний для захисту і ніде не задокументований. Для зломщика аналіз

такого коду можна порівняти з вивчення додаткової мови програмування. Це значно збільшує час на аналіз коду. Тому захист за допомогою віртуалізації є найбільш стійким.

Будь-який програмний додаток можна описати як набір значень які дозволяють досягнути потрібного результату. При цьому набір самих значень, їх порядок, кількість чи ефективність переходять на задній план, оскільки головний в даному випадку результат. Тобто роботу деякої функції:

$$F\ (p1, p2, p_n),$$

де F – функція, p – деякий параметр (їх кількість може бути різною, у тому числі й рівна нулю), можна описати за формулою:

$$(I_1 - p_1) + I_2\hat{}I_3 + I_n = R,$$

(операції між інструкціями I випадкові та написані лише для прикладу) де I – це деяка інструкція, яка виконує певні дії з вхідними параметрами p, якщо вони присутні, й іншими інструкціями, та R – деякий результат роботи, який повинен бути незмінним після обфускації коду при ідентичних умовах перевірки безліч разів (вхідні параметри, тощо). З цього виходить що при перетворюваній функції

$F\ (p1, p2, p_n)$ з тілом $(I_1 - p_1) + I_2\hat{}I_3 + I_n = R,$

результат перетворювання тіла:

$$(I_1 * p_1) + I_2 - (I_3/p_2) + I_n = R,$$

при умові що R, при ідентичних умовах залишається незмінним, вважається прийнятним. Отож умова успішного перетворення функції виглядає так:

$$I_n + p_n = R = I'_n + p_n,$$

при цьому захист програмного коду при деякій функції $F\ (p1, p2\ \ldots\ p_n)$ з іменем x, досягається за допомогою деякої функції $F_p(x)$, математична модель якої виглядає наступним чином:

$$F_p(I_n + p_n) = I'_n + p_n = x',$$

при умові що при будь-якій кількості будь-яких p_n буде виконуватись рівність:

$$x(p_n) = x'(p_n) = R,$$

де p_n – деяка кількість, деяких вхідних параметрів, x – деяка функція, яку необхідно захистити, x' – перетворена функція x, I_n –деяка інструкція, R –результат викання функції.

Розглянувши переваги та недоліки існуючих засобів захисту програмного забезпечення, було вирішено взяти за основу принцип перетворювання коду, тобто віртуалізацю. Це було обумовлено тим що системи захисту на його основі вважаються найбільш стійкими до зворотної інженерії. Перш за все було вирішено позбутися явної ідентифікації методу який викликається. Друга «слабкість» існуючих систем захисту це відсутність можливості контролю роботи. В час коли

інтернет є невід'ємною частиною майже будь-якого ПК, можна зробити висновок що розробка системи захисту програмного забезпечення на базі перетворювання коду з контролем переходів шляхом контролюючого серверу вважається виправданим. Таким чином за мету взято створити програмний додаток-обфускатор на базі перетворення коду та додаток-сервер в ролі «чорна скринька» для можливості віддалено контролювати роботу програми та відключити її по можливості. Системи захисту на базі віртуалізації мають ряд недоліків які будуть вирішені наступними кроками:

1. Було вирішено позбутися явної ідентифікації методу який викликається. Дана властивість притаманна всім відомим системам захисту. Цей недолік дозволяє аналізувати код статичним на комбінованим методами. Таким чином для ідентифікації необхідного для виконання коду необхідний запуск самої програми, що не є безпечно з точки зору розгляду програми код якої працює як за принципом «чорного» або «сірого» ящика.

2. Наступна «слабкість» існуючих систем захисту це відсутність можливості контролю роботи зі сторони менш захищеної системи в даному випадку це клієнтський додаток який може використовуватися в будь-якому оточенні. да В час коли інтернет є невід'ємною частиною майже будь-якого ПК, можна зробити висновок що розробка системи захисту програмного забезпечення на базі перетворювання коду з контролем переходів шляхом контролюючого серверу вважається виправданим. Таким чином за мету взято створити програмний додаток-обфускатор на базі перетворення коду та додаток-сервер в ролі «чорна скринька» для можливості віддалено контролювати роботу програми та відключити її по можливості.

Література

1. Binary Obfuscation Using Signals [Електроннийресурс]. – Режим доступу: http://static.usenix.org/event/sec07/tech/full_papers/popov/popov_html
2. Тим Джонс. Виртуализация приложений: История появления и перспективы дальнейшего развития [Електроннийресурс]. – Режим доступу: https://www.ibm.com/developerworks/ru/library/l-virtual-machine-architectures/index.html