УДК 004.4 http://doi.org/10.15407/pp2022.03-04.376

ВИЯВЛЕННЯ БОТНЕТ-ТРАФІКУ НА ОСНОВІ ПОТОКІВ, ВИКОРИСТОВУЮЧИ ШІ

Богдан Панчук

В даній роботі описано узагальнюючий підхід для побудови повноцінних систем для виявлення мережевої активності ботнетів методами штучного інтелекту (ШІ). Розглядається алгоритм реконструкції потоків мережевого трафіку як основний метод виділення числових характеристик, а також різні типи класифікаторів ШІ з метою досягення найкращої якості виявлення. Також були залучені результати робіт інших авторів у даній області. Компоненти описаної системи виявлення втручань були реалізовані та протестовані на публічно доступному наборі даних з відбитками трафіку справжніх ботнетів. У ході роботи було отримано та детально проаналізовано показники ефективності виявлення для різних моделей ШІ. Було застосовано і описано різні методи попередньої обробки даних, що допомогло ще більше покращити результати. Також пропонуються деякі варіанти для майбутніх удосконалень підходу до виділення мережевих характеристик. У кінці було здійснено порівняння даної реалізації з результатами, отриманими іншими дослідниками в даній області.

Ключові слова: інформаційна безпека, виявлення втручань, ботнети, мережеві потоки, штучний інтелект

This paper outlines the generalized framework for building end-to-end botnet network activity detection systems using artificial intelligence (AI) techniques. The paper describes network flows reconstruction as a primary feature-extraction method and considers different AI classifiers for achieving a better detection rate. The results of the latest research by other authors in the field are incorporated to implement a more efficient approach for botnet discovery. The described intrusion detection pipeline was tested on a dataset with real botnet activity traces. The performance metrics for different AI classification models were obtained and analyzed in detail. Different data preprocessing techniques were tried and described which helped improve the results even further. Some options for future enhancement of network feature selection were proposed as well. The comparison of the obtained performance metrics was drawn against the results provided by other researchers in this field.

Keywords: Information security, intrusion detection, botnet, network flow, artificial intelligence

Введення

У середовищі загально розповсюдженої інтеграції систем в мережу Інтернет мережеві атаки стали звичним явищем. Звичайні засоби захисту, як-от ретельно налаштовані правила брандмауера та антивірусні програми на основі сигнатур хоч і корисні, однак не охоплюють весь простір можливих атак. Безліч атак проводяться через загальновідомі відкриті порти і маскуються під звичайну мережеву активність. Таку активність неможливо виявити, використовуючи лише статично орієнтовані механізми безпеки, такі як брандмауери та методи співставлення сигнатур. Системи виявлення втручань (IDS, Intrusion Detection Systems) — це динамічна лінія захисту, яка призначена для виявлення шкідливої діяльності програм під час їх виконання. У цій статті описується мережева система виявлення втручань з використанням моделей ШІ для класифікації трафіку. Щоб краще зрозуміти переваги такого підходу, варто розглянути інші поширені альтернативи.

Ерве Дебар 2009 року [1] подав огляд різних типів IDS та їхню таксономію, розділивши їх за способами виявлення на базі знань і поведінки.

Системи з виявленням на базі знань (також відомі як «засновані на правилах») вважаються класичним підходом. Вони використовують набір заздалегідь визначених правил, створених людиною-експертом, що описують відомі моделі атак і порівнюють їх із зафіксованими подіями. Події можуть можуть мати різні представлення: рядки журналів роботи застосунку (логи), аудиту системи, звичайні мережеві пакети та інші. Прикладом такого правила може бути *«сталося більше, ніж N невдалих спроб підключення з IP-адреси X»*. Після спрацювання якогось із правил система генерує сповіщення і потенційно виконує автоматичну відповідь (наприклад, блокує шкідливу IP-адресу). У такого підходу є кілька недоліків. По-перше, для створення правил і подальшого оновлення їх списку потрібна людина-експерт. По-друге, побудова комплексних правил може бути надзвичайно складною, оскільки моделі атаки не завжди можна легко описати. Зловмисники часто навмисно розтягують втручання в часі, надсилаючи шкідливі пакети нерегулярно, а часто вперемішку з доброякісними даними. Врешті решт, як випливає з назви, системи, що базуються на знаннях, вимагають спеціальних попередніх відомостей про атаку для її виявлення.

Системи, засновані на поведінці, натомість, є більш гнучкими, оскільки призначені для «вивчення» поведінки користувача та створення так-званого «профілю нормальної активності». Відповідно події, що вибиваються з «нормального» шаблону активності (аномалії), класифікуються як вторгнення. На жаль, незважаючи на вищу гнучкість, складність цього підходу полягає у складності самого процесу визначення «нормальної поведінки». Звичайні користувачі можуть із часом змінювати шаблони своєї звичної поведінки, і важко визначити, наскільки ця зміна прийнятна, перш, ніж вважати її аномалією.

Системи, керовані штучним інтелектом, є альтернативою, яка виглядає найбільш перспективно, враховуючи швидкий розвиток ШІ за останнє десятиліття. Перевага штучного інтелекту полягає в тому, що він містить сильні сторони як систем заснованих на знаннях, так і систем на базі вивчення поведінки, водночас по-

требуючи значно менше зусиль від інженера-розробника для опису та налаштування. Моделям ШІ не потрібні люди-експерти, оскільки вони самі вивчають правила та профілі з існуючих даних. Наприклад, дерева рішень та нейронні мережі можуть у процесі тренування виводити значно складніші набори правил порівняно із системами виявлення на базі знань. Моделі машинного навчання «з учителем» вивчають нормальний профіль активності користувача одночасно з вивченням шаблонів самих атак, а моделі «без учителя» можуть самостійно навчитися відрізняти аномальний трафік, що робить їх досконалішими за системи виявлення на базі поведінки.

Однак, системи, керовані ШІ, мають два основні недоліки: по-перше, виведені ними правила, часто не піддаються інтерпретації людиною; по-друге, потрібен великий набір навчальних даних – у випадку ж навчання моделей «з учителем» (supervised learning) набір тренувальних даних має бути ще й промаркований (тобто, там мають міститися не лише вхідні дані, а й вірний очікуваний вихідний результат).

Дослідження показали, що якість IDS на основі штучного інтелекту значно залежить від методу виділення ознак, а також від вибору базової архітектури класифікатора. У роботі під редакцією Аль-Сакіба Хана Патана 2014 року [2] коротко описані як переваги, так і недоліки використання різних методів ШІ для виявлення. Там згадуються: глибокі нейронні мережі (DNN), самоорганізаційні карти Кохонена, моделі Маркова, байєсівські системи та машини опорних векторів (SVM). Більш детально, Arnaldo et al. 2017 року [3] описав використання випадкових лісів (RF), нейронних мереж із прямим зв'язком (FFNN), згорткових нейронних мереж (CNN), а також рекурентних нейронних мереж (RNN) для аналізу журнальних та реляційних даних, представлених у вигляді часових рядів. У тій роботі, однак, ні RNN, ні CNN не покращили результати в порівнянні зі звичайними FFNN або RF. У межах цієї ж статті буде показано, що LSTM (варіація RNN) і CNN таки здатні покращити результати аналізу мережевих потоків за умови, якщо попередня обробка та агрегація даних виконані добре.

У цій статті головна увага зосереджується на виявленні саме трафіку ботнетів. Ботнети є одними з найбільш руйнівних загроз кібербезпеки, і їхня кількість щороку активно зростає. На відміну від інших атак, ботнети створюються так, щоб залишатися прихованими протягом тривалого періоду часу – чим довше інфекція залишається невиявленою, тим більше шкоди завдається у вигляді витоку даних, зловмисного спостереження та мимовільної участі зараженої машини в інших мережевих атаки. Саме тому завчасне виявлення мереж ботів життєво важливе для будь-якої системи інформаційної безпеки.

Слід зазначити, що, хоча фокус статті зроблений на ботнетех, однак спеціалізація загальної схеми проводиться лише на етапах виділення числових характеристик з мережевого трафіку, а також вибору наборів даних, що використовується для навчання моделей ШІ. Загальну ж систему виявлення можна застосувати і для інших видів мережевих атак, а відмінність буде лиш у тренувальному датасеті та обраній множині мережевих характеристик.

Виявлення втручань на основі мережевих потоків

У цьому розділі наведено високорівневий опис функціональної лінії обробки даних для виявлення втручань на основі мережевих потоків. Кожен етап буде детальніше описано в наступних розділах.

Рис. 1 демонструє загальний алгоритм для виявлення втручань за допомогою ШІ.

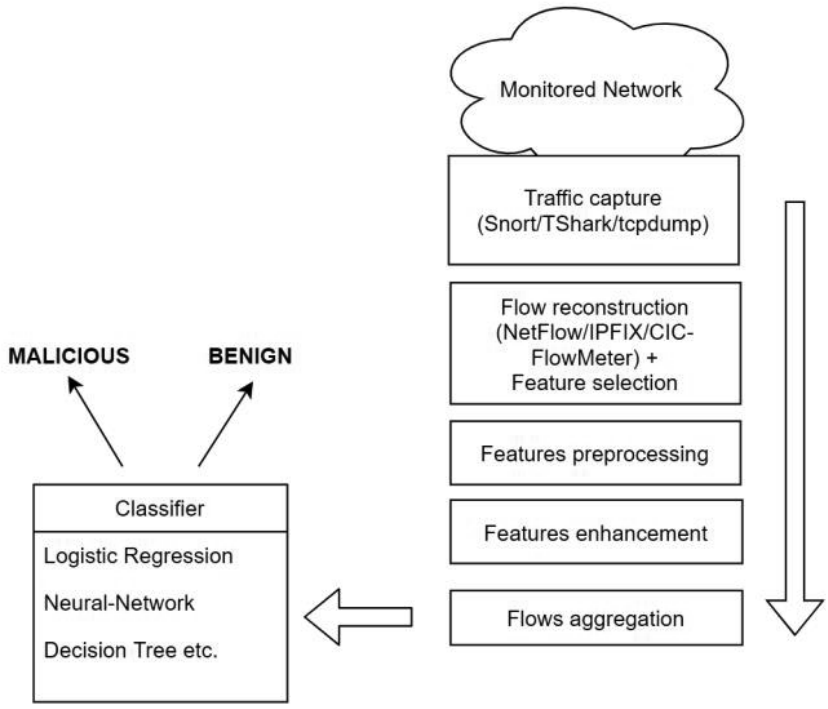

Рис. 1. Функціональна лінія виявлення втручань на основі штучного інтелекту

Початковий етап – це моніторинг мережі у вигляді захоплення трафіку. Захоплення має відбуватися на важливих вузлах мережі, за якою ведеться спостереження. Захоплені пакети передаються далі по лінії обробки для їх подальшого аналізу.

Другий етап – реконструкція мережевих потоків із захоплених пакетів. Мета мережевого потоку – представити наскрізний діалог між двома віддаленими процесами (кінцевими точками). Основна перевага такого підходу полягає в тому, що він дозволяє ізолювати групи пакетів, які є частиною одного комунікативного процесу, від решти непов'язаного трафіку. Таким чином, незалежно від того, скільки шуму проходить через канал, класифікатор може оперувати з окремими діалогами.

Ще однією перевагою роботи з мережевими потоками є легке вилучення контекстних ознак, що є третім етапом лінії обробки. Для всього масиву пакетів обчислюються сукупні значення, – це можуть бути статистичні значення, що описують такі властивості пакетів, як частота, розмір тощо. Також можна враховувати виникнення деяких мережевих подій у потоці (наприклад, наявність спроби резолюції доменного імені). Ці числові характеристики описують повний контекст комунікації. Впродовж певного проміжку часу між двома віддаленими процесами може проходити не один, а кілька потоків. Тому важливо зафіксувати часову мітку, коли поточний діалог був ініційований і коли він закінчився – її можна буде використовувати на подальших етапах лінії обробки.

Після отримання чисельного представлення кожного окремого потоку четвертим кроком є попередня обробка отриманих числових характеристик: відсіювання недійсних значень, нормалізація та масштабування. На цьому етапі вже можливо виконувати класифікацію трафіку на базі лише цих отриманих характеристик. Однак, як буде показано далі, ефективність виявлення можна підвищити шляхом покращення ознак через застосування таких методів, як зменшення розмірності та розумного вибору характеристик. Цей етап є необов'язковим.

Успішність виявлень можна підняти навіть вище, якщо замість того, щоб аналізувати кожен потік окремо, об'єднати їх в упорядковані групи та розглядати як часові ряди. Цей підхід був запропонований Arnaldo et al. 2017 року [3], хоча в тій роботі він описується лише в межах групування пов'язаних подій за фіксованими часовими інтервалами та обчислення статистичних характеристик для кожної групи. Така агрегація на основі часових проміжків не еквівалентна техніці реконструкції мережевих потоків. Більш детальна інформація наведена в наступному розділі.

Останнім етапом є класифікація потоків моделлю ШІ на основі ознак, отриманих на попередніх етапах. Залежно від архітектури моделі, може здійснюватися класифікація як окремих потоків, так і цілих серій.

Власне, проблема мережевого виявлення зводиться до агрегації захоплених пакетів у потоки, що представлені векторами числових (або категорійних) ознак, а потім розмарковування їх на шкідливі та доброякісні за допомогою моделі класифікації на основі штучного інтелекту.

Цей підхід передбачає навчання «з учителем», тому для навчання моделі необхідно надати розмічений набір даних. Такий набір може бути створений в лабораторних умовах шляхом штучного генерування шкідливого трафіку (наприклад, інфікуючи машини ботнетом) і подальшого захоплення їх мережевої активності. Маркування виконується, апріорі знаючи зловмисні IP-адреси та часові інтервали – після реконструкції потоків, усі ті, які мають задані IP-адреси як своє джерело чи пункт призначення (і підпадають під відповідні часові інтервали, якщо вони вказані), вважаються шкідливими. Інші потоки позначаються як доброякісні.

В межах створення цієї статті було реалізовано повний алгоритм виявлення описаний вище: розмічену множину потоків отримано з датасету ISCX Botnet 2014 [4], наданого Канадським інститутом кібербезпеки. Далі виконано обробку та агрегацію отриманих ознак. Згодом на основі цих числових ознак були натреновані різні моделі класифікації ШІ. Нарешті, моделі були протестовані на небачених раніше даних і були обраховані показники продуктивності для кожної з них. Більш детальна інформація наведена в наступних розділах.

Захоплення мережевого трафіку

Захоплення подій є важливою частиною кожної системи виявлення втручань. У випадку системи виявлення, описаної вище, у ролі подій виступають мережеві пакети. Моніторинг мережевого трафіку зазвичай виконується через «сніффінг пакетів». Ця група методів дозволяє захоплювати мережеві пакети, що проходять через мережеві інтерфейси пристроїв у будь-якому напрямку, на рівнях 2-5 моделі OSI. Звичайними інструментами для виконання таких завдань є:

– Wireshark – найпоширеніший інструмент захоплення пакетів із користувацьким інтерфейсом і можливістю виконувати різноманітну фільтрацію (наприклад, на основі IP-адреси джерела/призначення, протоколу, порту, тощо). Він може розпізнавати різні протоколи на різних мережевих рівнях і відтворювати пакети, сегменти, запити та інше.

– TShark – консольна версія Wireshark. Може працювати як фоновий процес і є найбільш корисним у середовищах з багатьма хостами, налаштування яких було автоматизовано.

Однак, слід зазначити, що аналіз звичайних пакетів є доволі дорогим та іноді недоступним процесом. По-перше, через величезну кількість даних, що генеруються: по суті захоплення пакету означає його дублювання на якийсь інший хост перш, ніж він передасться аналізатору. Це принаймні вдвічі збільшує обсяг даних, що передаються та зберігаються. Поза тим, буває неможливо використовувати звичайні інструменти захоплення на певних мережевих пристроях (наприклад, на деяких комутаторах і маршрутизаторах), без погіршення їх пропускної здатності. Такі методи як «дзеркальне відображення портів» подвоюють об'єм трафіку і підтримуються не на всіх мережевих пристроях (особливо дешевих). Більше того, недостатньо захоплювати пакети на одному вузлі мережі:

пакети з однаковими пунктами призначення можуть подорожувати різними маршрутами через мережу, тому для захоплення всіх можливих пакетів необхідно також спостерігати за всіма проміжними вузлами. Це, однак, призводить до того, що один і той самий пакет може бути перехоплений на кількох хостах, через які він пройшов, що, в свою чергу, призводить до ще більшого дублювання даних. Усі ці великі обсяги даних потрібно зберігати та аналізувати, часто в режимі реального часу, що може бути неприпустимо витратним з точки зору обчислень. Нарешті, дублювання даних і аналіз необроблених пакетів завжди є ризиком для безпеки, оскільки, цілком ймовірно, вони містять конфіденційну інформацію. Крім того, з точки зору IDS, хоч тіло пакету і може містити важливу інформацію для ідентифікації атаки (наприклад, команди комунікацій IRC-ботнету), але глибокий аналіз пакетів стає неможливим, коли трафік зашифрований (що часто й трапляється).

Реконструкція мережевих потоків

Доповнюючим підходом до аналізу мережевого трафіку, який вирішує проблеми, згадані вище, є обробка цілих мережевих потоків, а не окремих пакетів. Щоб досягти цього, потоки потрібно реконструювати із окремих пакетів і лише потім передати в аналізатор.

Існують різні способи реконструкції мережевих потоків, різної складності та якості результату. В теорії мережевий потік має об'єднувати всі пакети, що мають один і той самий контекст та часовий інтервал, передані між двома процесами (кінцевими точками), для відтворення повного сеансу комунікації між ними. Прикладом може бути послідовність пакетів, обмін якими відбувається в межах одного TCP-з'єднання. Потоки можуть бути однонаправленими або двонаправленими, залежно від процедури реконструкції та вимог. Як випливає з назви, однонаправлений являє собою набір пов'язаних пакетів, що надходять від однієї кінцевої точки до іншої в одному напрямку, тоді як двонаправлений включає пакети, що йдуть в обох напрямках. Мережевий потік ідентифікується наступним кортежем значень:

<IP-адреса джерела, IP-адреса точки призначення,
Порт джерела, Порт точки призначення,
(Транспорт) Протокол, Час початку (Час закінчення)>

Отже, маючи набір мережевих пакетів, потік можна розглядати як агрегат пакетів, згрупованих за цими атрибутами. Агрегація виконується шляхом обчислення набору значень, які нас цікавлять (далі будуть називатися ознаками/характеристиками), що представляють узагальнений опис властивостей усіх пакетів, пов'язаних з даним потоком. Ці характеристики можуть бути статистичними (наприклад, середній/мін./макс. розмір пакету), категоріальними (наприклад, тип протоколу) або простими лічильниками (кількість конкретних TCP-флагів, загальна кількість пакетів, тощо). Усі ці значення надзвичайно важливі і в подальшому будуть використані системою виявлення для прийняття рішення про те, чи є потіки/серія потоків шкідливими, чи ні.

Алгоритм реконструкції потоку можна розділити на 3 основні процеси: 1) реєстрація нових активних потоків 2) співставлення пакета з одним із поточно-активних потоків 3) маркування потоків, які досягли однієї з умов термінації, як завершені, та їх видалення зі списку активних потоків. Алгоритм постійно відстежує список вже виявлених активних потоків; для кожного захопленого мережевого пакета перевіряється, чи є активний потік, з яким цей пакет може бути пов'язаний. Якщо пакет відповідає одному з активних потоків, значення характеристик цього потоку переобчислюються відповідно. Якщо не знайдено жодних активних потоків, з якими можна пов'язати пакет, створюється новий потік, який додається до списку активних потоків. Згаданий вище набір атрибутів пакета використовується для однозначної ідентифікації потоку. Оскільки це перший пакет у потоці, його часова мітка поміщається в поле «Час початку» потоку. Цей процес повторюється для кожного пакета, що аналізується. Крім того, потоки іноді завершуються та видаляються зі списку активних потоків. Список активних потоків потрібно періодично перевіряти на наявність потоків із вичерпаним терміном дії, або за «тайм-аутом простою», або за «тайм-аутом активності» – попередньо визначеними зовнішніми параметрами, які контролюють тривалість життя потоку:

– тайм-аут активності потоку – максимальний інтервал часу, впродовж якого потік вважається активним. Якщо поточний час життя потоку перевищує це значення, він позначається як завершений і створюється новий потік. Припинення потоку за тайм-аутом активності показано на Рис. 2.

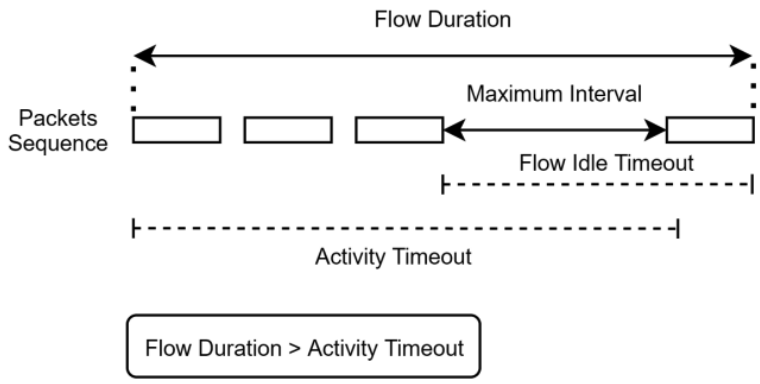

Рис. 2. Завершення потоку за тайм-аутом активності

– тайм-аут простою потоку – максимальний проміжок часу, дозволений між двома послідовними пакетами в одному потоці: якщо дозволений час простою минув, але новий пакет в потоці не зареєстровано, то потік вважається завершеним і видаляється зі списку активних потоків. На Рис. 3 показано завершення потоку через вичерпання дозволеного часу простою.

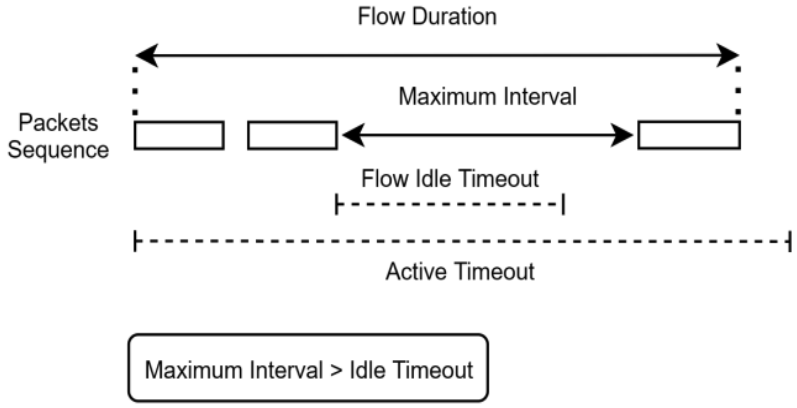

Рис. 3. Завершення потоку за тайм-аутом простою

Крім параметрів тайм-аутів, TCP-з'єднання мають деякі додаткові умови завершення (див. Рис. 4):

– Найпоширенішою є послідовність бітових прапорців «ввічливого завершення з'єднання»: FIN-ACK-FIN-ACK. Якщо зустрічається перший прапорець FIN, алгоритм шукає наступні прапорці ACK-FIN-ACK, а потім позначає потік як завершений.

– Інший випадок завершення потоку TCP – це коли зустрічається прапорець RST, який означає, що з'єднання було грубо перервано одним із учасників.

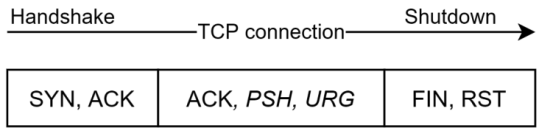

Рис. 4. життєвий цикл TCP-з'єднання та відповідні бітові прапорці

Для UDP і ICMP потоків тайм-аути активності та простою є єдиними умовами завершення, оскільки інших механізмів завершення з'єднання вони не мають.

Формати представлення потоку

На практиці для побудови лінії аналізу мережевих потоків з метою виявлення втручання, важливо обрати формат для представлення числових характеристик потоку, а також відповідний інструмент для виділення цих значення.

Марком Гремом 2018 року [5] було проведено фундаментальну роботу з аналізу форматів потоків та вибору ознак. У його праці описано два основні представлення потоків, якими є NetFlow (різних версій) та IPFIX (IP Flow Information eXport).

NetFlow — це технологія, розроблена та запатентована компанією Cisco 1996 року та вбудована в більшість маршрутизаторів компанії. Перша реалізація (V1) обмежувалась трафіком IPv4 і містила лише кілька полів для опису потоку. Згодом, після кількох внутрішніх ітерацій, була випущена версія V5, яка зараз є найбільш вживаною разом із V9. V5 має 18 статичних полів із заголовками, що характеризують потік, включно.. На жаль, лише 10 є корисними для виявлення втручань. Остаточною версією NetFlow є V9, в якій була розширена кількість статичних полів до 79 (104 для пристроїв Cisco) і було додано підтримку шаблонів. Це дозволяє налаштовувати формат зібраної інформації про потоки відповідно до потреб: деякі поля можна додати, а інші можна виключити.

Існують дві основні проблеми з NetFlow в контексті IDS: 1) він є запатентованим 2) з самого початку він не розроблявся як інструмент моніторингу трафіку для виявлення втручань, а радше для спостереження за споживанням мережевих ресурсів та аналізу продуктивності або ж просто для керування мережею.

IPFIX (він же NetFlow V10, хоча й не має нічого спільного з оригінальним Cisco NetFlow) — це стандартизований протокол/технологія для захоплення мережевих потоків. Він має 433 інформаційних елементи, 79 з яких відповідають NetFlow версії 9 для зворотньої сумісності. Як і NetFlow V9, вона базується на шаблонах, тому з цих 433 полів користувач може сам обрати множину полів для захоплення та відправки. Як у випадку V9, технологія підтримує IPv6, багатоадресну передачу та MPLS. IPFIX не залежить від виробника і відповідає стандартам RFC 7011. Однією з основних відмінностей IPFIX від NetFlow є те, що IPFIX також підтримує аналіз протоколу прикладного рівня (L7) (як-от, HTTP та IRC), що є дуже

корисним для виявлення загроз. За словами Марка Грема, 2018 [5] IPFIX є кращим форматом для виявлення втручань.

Однак, хоча NetFlow та IPFIX забезпечують якісне та широке представлення потоків, потоки в їхньому форматі виявилось важко отримати з існуючих наборів даних (датаестів). Для обробки датасета ISCX Botnet 2014 [4], який надається у вигляді PCAP-файлів (Wireshark-сумісний набір захоплених пакетів), потрібен був інструмент для перетворення пакетів із цих файлів у потоки (чи у форматі IPFIX, чи у будь-який інший). І хоча IPFIX є відкритою технологією, однак більшість інструментів, здатних створювати потоки у форматі IPFIX із наборів пакетів, є пропрієтарними (наприклад, Nprobe). Отже, для реконструкції потоків та виділення первинних ознак в межах цієї роботи було обрано CICFlowMeter V4.0 [6] — інструмент із відкритим вихідним кодом, розроблений Канадським Інститутом Кібербезпеки. Він може конструювати потоки з пакетів як в режимі офлайн (з файлів PCAP), так і в режимі реального часу (шляхом сніффінгу трафіку). Вихідні потоки представлені у форматі CSV. Кожен побудований потік містить 83 інформаційних поля.

Виділення характеристик (ознак) потоку

Із 83 полів потоку, отриманих через CICFlowMeter, 6 полів (а саме *IP-адреса джерела/місця призначення*, *порти джерела/місця призначення*, *протокол*, *часова мітка початку*) були використані для ідентифікації потоку. Для класифікації було використано ще 72 поля характеристик, а саме: *Flow Duration, Fwd/Bwd Header Length, (Fwd/Bwd) Packet Length Min/Max/Mean/Std/Total, Total Fwd/Bwd Packets, (Fwd/Bwd) Inter-Arrival Time Min/Max/Mean/Std/Total, (Fwd/Bwd) SYN/FIN/ACK/RST/CWR/PSH/URG/ECE flags count, Packets/second, Bytes/second, Flow Active Duration Min/Max/Mean/Std, Subflow (Fwd/Bwd) Packets/ Bytes, Up/Down Ratio*.

Решта полів залишилися невикористаними через помилки, які систематично виникали при їх обчисленні даним інструментом.

Примітка: оскільки потоки вважалися двонаправленими, *Fwd* - стосуються пакетів, що надіслані у напрямку від хоста, який ініціював потік, а *Bwd* - стосується пакетів, що надіслані у зворотньому напряму відповідно.

Попередня обробка характеристик

На цьому етапі вже можна натренувати модель класифікації потоків на основі полів-характеристик, отриманих на попередньому етапі. Однак для отримання прийнятних результатів необхідно здійснити додаткову попередню обробку даних.

По-перше, важливо очистити дані. На практиці часові мітки пакетів не завжди відповідають порядку їх надходження та обробки, що призводить до неправильного обчислення значень характеристик (у разі, якщо це не було передбачено інструментом реконструкції потоків). Наприклад, після обробки набору даних ISCX Botnet 2014 за допомогою CICFlowMeter для деяких потоків певні поля, як-от *Flow Duration (тривалість потоку)* та *Inter-Arrival Time (час між прибуттям)*, мали негативні значення. Такі потоки були відкинуті як недійсні.

По-друге, велику увагу необхідно приділяти нормалізації та стандартизації даних. Через природу обраних характеристик їнє значення можуть дуже відрізнятися – як у порівнянні з самими собою (у різних потоках), так і один з одним (у межах одного потоку). Наприклад, тривалість потоку може варіюватися від кількох мілісекунд до кількох годин. Така неоднорідність негативно впливає на здатність моделі навчатися. Щоб компенсувати це, дані потрібно нормалізувати. У цій роботі, для кожної характеристики було використано мінімальне і максимальне масштабування (1), тому їхні кінцеві значення знаходяться в діапазоні від 0 до 1.

$$x_{std} = (x - x_{min})/(x_{max} - x_{min}), \quad (1)$$

Крім того, значення деяких лічильників (наприклад, лічильники кількості TCP-флагів) на кілька порядків менші за значення інших ознак потоків, і тому вони можуть бути «поглинуті» полями із суттєво більшими значеннями, як-от, *довжина пакета* або *тривалість потоку*. Ця проблема вирішується стандартизацією ознак шляхом віднімання математичного сподівання та масштабуванням за стандартним відхилення кожної ознаки. У випадку класифікатора логістичної регресії така маніпуляція допомогла покращити показник виявлення AUROC (Area Under ROC Curve) з 0,778 до 0,7997.

Можливі удосконалення у виборі ознак

Варто зазначити, що, хоча використання статистичних характеристик для узагальненого опису мережевого потоку дає досить високі показники виявлення (AUROC 0,918 на незнайомий даних), створення інших характеристик (ознак), специфічних для виявлення ботнетів, може значно поліпшити результати.

Використовуючи кластерний та кореляційний аналіз на даних, зібраних з реальних ботнетів, Марк Грем, 2018 року [5] дійшов висновку, що ознаки, наведені в Табл. 1, найбільше сприяють успішному виявленню їхньої активності. Незважаючи на те, що деякі поля є суто утилітарними й очевидними (наприклад, IP-адреси), інші, як-от, прапорці L7 (DNS, HTTP, SMTP, IRC), тісно пов'язані зі способами, через які у ботнетах відбувається комунікація. Як приклад, відомі методи «Fast flux» [11] і «Algorithm Generation Domain» (DGA)

[12], що використовують DNS-запити для знаходження бот-мастера — вузла-центра командування та керування (Command and Control, C&C). Вивчення особливостей функціонування ботнетів, шаблонів їх мережевого трафіку та притаманних їм евристик може допомогти обирати набори ознак для найбільш ефективного виявлення. Це саме твердження дійсне і для виявлення інших типів мережевих атак.

Таблиця 1. IPFIX-шаблон полів потоків для виявлення ботнетів [5]

Назва поля	Опис
srcIPv4	IPv4 джерела
dstIPv4	IPv4 пункту призначення
srcPort	Порт джерела
dstPort	Порт пункту призначення
packetTotal	Загальна кількість переданих пакетів
flowEndMS	Часова мітка кінця потоку
flowStartMS	Часова мітка початку потоку
protocol	OSI протокол (TCP, UDP, ICMP і т.д.)
initTCPFlag	Прапорці, передані під час встановлення TCP-з'єднання
tcpSeqNos	Номери послідовності в TCP з'єднаннях
collectorIPv4	IPv4 адреса колектора потоків
flowKeyHash	Хеш кортежу, який ідентифікує потік
ircTextMessage	Текст та команди у випадку комунікації через IRC
httpGet	Наявність HTTP GET запиту
httpResponse	Код відповіді HTTP
dnsARecord	У потоці міститься резолюція домену записа типу «A»
dnsSOARecord	У потоці міститься резолюція домену записа типу «SOA»
smtpHello	В потоці містяться SMTP-команди «ELHO» чи «HELO»
sslName	Деталі центру SSL-сертифікації

Методи класифікації мережевих потоків

Існує кілька різних способів для класифікації захоплених слідів мережевої активності. Основною класифікаційною одиницею є сам мережевий потік. Оскільки (під час навчання) кожен потік вже промаркований як шкідливий або доброякісний, вектори ознак потоків можна подавати в класифікатор один за одним, щоб на основі закодованих у них значень, дізнаватися, чи є потік частиною комунікацій ботнету. Якщо кожен вектор потоку містить F ознак, розмірність вхідних даних класифікатора та розмірність повної вхідної матриці під час навчання дорівнюватиме N×F, де N — кількість потоків у навчальному наборі. Цей метод буде використовуватися як базовий для подяльших порівнянь. Найбілш продуктивними у такому підході виявились глибокі нейронні мережі (DNN) з 3 прихованими шарами 16x16x16 (AUROC - 0,866). Класифікатор «випадковий ліс» (Random Forest) посів друге місце (AUROC – 0,848), а логістична регресія – третє (AUROC – 0,778).

Другий, складніший підхід до класифікації, зосереджений на ідеї обробки наборів потоків, представлених у формі часових рядів і аналізу кожного нового потоку в контексті попередніх. Для досягнення цього необхідно враховувати часові проміжки кожного потоку – всі потоки відсортовуються на базі часових міток їх початку, і таким чином із окремих потоків утворюються часові ряди. Після цього можна використовувати різні методи для обробки таких рядів. Потоки можна розділити на групи за фіксованими часовими інтервалами або ж на групи з однаковою кількістю потоків у кожій. Пізніше ці групи об'єднуються в один вектор, який і передається у класифікатор. Однак, групування незв'язних між собою потоків, може давати нечіткі результати. Тому, окрім групування лише на основі їх послідовності у часі, доцільно також об'єднувати потоки на основі їхніх внутрішніх характеристик, таких як пара IP-адрес джерело-пункт призначення, портів та протоколів. Оскільки порт і протокол зв'язку зловмисник може змінювати досить регулярно, то щоб охопити повний процес комунікацій між ботом та C&C, мережеві потоки найкраще згрупувати саме за IP-адресами джерела та пункту призначення. Крім того, класифікація здійснюється для всієї групи потоків, а не лише для окремого (ізольованого) потоку, а група розглядається як часовий ряд. Об'єднавши ці два підходи, ми можемо робити висновки про характер всієї комунікації, оскільки буде враховано стан кожного потоку як у часі, так і в просторі подій. У цьому випадку класифікатор

приймає вхідні дані як матрицю M×F, де F — кількість ознак, а M — розмір групи потоків. Вхідний навчальний тензор K×M×F, де K — це кількість груп. Для класифікації часових рядів використовуються згорткові нейронні мережі з темпоральними згортками. Такий підхід був використаний Arnaldo et al. 2017 року [3], де описано його застосування для даних отриманих з журналів подій (історії з'єднанння), а також для даних мережевих потоків, отриманих з датасету ISCX Botnet 2014. У їхньому випадку, однак, це не принесло суттєвих результатів: якість класифікації була гіршою, ніж у випадку аналізу індивідуальних потоків. Хоча їм таки вдалося дещопокращити результат (на 5,6%) після доповнення початкового набору ознак додатково згенерованими.

Під час цієї роботи був використаний подібний підхід, але з деякими модифікаціями та покращеннями. По-перше, базовий набір ознак для кожного потоку був розширений з 23 до 72, що робить вхідні дані репрезентативнішими. По-друге, групування було проведено лише на основі IP-адреси джерела потоку. Це зроблено з метою визначення інфікованого хоста, який за метою знаходження C&C надсилає запити на різні віддалені IP-адреси, намагаючись знайти доступну. Спроби пошуку та встановлення з'єднання з центром командування та контролю притаманні для всіх ботнетів. Остання відмінність у тому, що в методі використаному Arnaldo et al. [3] розглядаючи групи з більш, ніж одним потоком (N = 7, 14, 28) відкидалися групи, котрі містять менше, ніж N потоків, а тому для різних випадків вони тренували та використовували різні моделі класифікаторів. У межах нашої статті, на противагу, було обрано розмір статичної групи (N = 10), і для всіх груп, з меншою кількістю елементів, вхідний тензор був доповнений нулями, що є загальноприйнятою практикою, застосовною при використанні згорток на групах даних неоднорідної розмірності. Разом з методами попередньої обробки та фільтрації, описаними в минулих розділах, використання конволюційної нейронної мережі (Convolutional Neural Network, CNN) для класифікації часових рядів дало значно кращі результати порівняно з базовою класифікацією окремих потоків звичайною нейронною мережею. Показники AUROC зросли на 6% (до 0,918), що свідчить про ефективність розгляду одразу кількох зв'язаних потоків із врахуванням їх порядку. Застосована CNN модель містила 2 згорткові шари і 2 одновимірних шари стягування. Перший згортковий шар мав розмір ядра 3 з 32 фільтрами, другий – розмір ядра 2 з 16 фільтрами. Цей підхід, однак, має серйозний недолік: його непрактично використовувати для класифікації в реальному (чи напівреальному) часі, оскільки він вимагає аналізу всіх потоків групи разом.

Інший широко відомий підхід до аналізу часових рядів заснований на обробці подій по черзі у порядку їх виникнення, але з урахуванням попередніх подій на кожному наступному кроці. Для цього спочатку генерується вектор «історичних значень ознак» через застосування моделі до події попереднього часового кроку (у нашому випадку через застосування до попереднього потоку в групі). Цей вектор представлятиме історичний контекст для потоку, що аналізується на даному кроці. Потім історичні ознаки комбінуються зі значеннями ознак поточного екземпляра і отриманий вектор передається класифікатору, який вже робить висновок на основі усіх ознак разом. Вхідні дані класифікатора можна описати як (2).

$$F_i = F_{historical} \cup F_{current} = (x^h_1, \dots, x^h_{|F_{historical}|}, x^c_1, \dots, x^c_{|F_{current}|}) \quad (2)$$

де $F_{current}$ - вектор ознак поточного екземпляра, та $F_{historical} = E(\bigcup^k_{current-1} F_k)$ – вектор історичних ознак, який, для зменшення кількості підрахунків, можна подати як (3).

$$F_{historical} = E(\bigcup^k_{current-1} F_k) = E(F_{historical-1} \cup F_{current-1}) \quad (3)$$

Такий підхід може бути реалізований за допомогою рекурентної нейронної мережі (Recurrent Neural Network, RNN), перевага якої полягає в тому, що серія векторів ознак оброблюється елемент за елементом. Групи потоків обробляються послідовно. Підхід відстежує залежності між подіями серії і рекурентно передає значення, отримані на попередньому кроці (історичні ознаки), на вхід класифікатора в даному кроці. Іншими словами, такі класифікатори можуть «запам'ятовувати» попередні стани та враховувати цей досвід під час виконання майбутніх класифікацій. У цій роботі в якості RNN була використана архітектура «довгої короткочасної пам'яті» (Long Short-Term Memory, LSTM), яка є більш стійкою до проблеми зникаючих градієнтів порівняно зі звичайною RNN. Показники AUROC досягли значення 0,907, що на 2,37% краще за звичайний аналіз окремих потоків, але все ж гірше за CNN.

Моделі класифікації

Проаналізуваши експериментальні результати якості різних моделей ШІ для виявлення ботнетів, було б корисно узагальнити причини успіхів чи невдач, а також виділити можливі випадки їх використання.

У ролі базового класифікатора для порівняння можна взяти логістичну регресію. Завдяки своїй простоті ця модель швидко навчається і має найкращу продуктивність при роботі в режимі реального часу. Крім того, отримані при навчанні вагові коефіцієнти можна додатково інтерпретувати розуміння, які саме ознаки потоків роблять найбільший внесок у процес ухвалення рішень. Результатом класифікації логістичною регресією є значення ймовірності, а регуляцію порогу ймовірності можна використовувати для зменшення кількості помилкових спрацьовувань у разі необхідності, що є вагомою перевагою для систем виявлення втручання. Логістичну регресію можна використовувати для децентралізованих IDS, що працюють з обмеженою ємністю ресурсів на кожному вузлі. Під час експериментів для класифікатора логістичної регресії показник ефективності AUROC склав 0,778, а точність виявлення – 0,78. Ці значення були поліпшені шляхом застосування стандартного масштабування (описано в Розділі 7), а також зменшенням розмірності вхідних даних за допомогою методу головних компонент (Principal Componet Analysis, PCA) (див. показники в Табл. 2).

Таблиця 2. Порівняння ефективності моделей ШІ на наборі даних ISCX Botnet 2014

Робота	Вибір ознаки та зменшення розмірності	Класифікатор	Показники ефективності (на незнайомих даних)
Ця робота.	PCA (16) (1 потік)	Логістична регресія	AUROC 0.804, Точність 0.78
	PCA (16) (1 потік)	Дерево прийняття рішень (глибина 9)	AUROC 0.844, Точність 0.833
	PCA (16) (1 потік)	Випадковий ліс (глибина 9)	AUROC 0.91, Точність 0.87
	(1 потік)	Логістична регресія	AUROC 0.778 (0.799), Точність 0.648 (0.722)
	(1 потік)	Дерево прийняття рішень (глибина 9)	AUROC 0.477, Точність 0.65
	(1 потік)	Випадковий ліс (глибина 9)	AUROC 0.848, Точність 0.798
	(1 потік)	DNN (16x16x16)	AUROC 0.866
	Групування за IP джерела (10 потоків)	CNN (2 конволюції + стяжка)	AUROC 0.918
	Групування за IP джерела (10 потоків)	LSTM (100 комірок)	AUROC 0.908
Arnaldo et al. [3]	PCA (1 потік)	Випадковий ліс	AUROC 0.769
	(1 потік)	Випадковий ліс	AUROC 0.768
	PCA (28 з'єднаних потоків) + згенеровані ознаки	Випадковий ліс	AUROC 0.811
	(1 потік)	DNN	AUROC 0.724
	Групування за IP джерела та призначення (7 потоків)	CNN	AUROC 0.644
	Групування за IP джерела та призначення (7 flows)	LSTM	AUROC 0.624
Meshal Farhan et al. [7]	MUTAL (1 flow)	Випадковий ліс	Точність 0.81
	MUTAL (1 flow)	Логістична регресія	Точність 0.748
	ANOVA (1 flow)	Дерево прийняття рішень	Точність 0.73
	MUTAL (1 flow)	DNN	Точність 0.736

Дерева рішень виявились одним з найкращих кандидатів для складної класифікації активності та виявлення втручань. Причина полягає в тому, що ознаки (характеристики) подій мережевої активності завжди мають певне семантичне значення і обираються, базуючись на знанні вже відомих методів та шаблонів атак. Ось кілька прикладів таких ознак для виявлення ботнетів на основі мережевих потоків: «чи був у потоці використаний протокол IRC?», «чи було зроблено DNS-запити типу A і скільки разів?», «яка кількість відмов (тобто була відсутня відповідь на прапорець SYN) і чи грубого розриву (зустрівся прапорець RST) відбулася під час TCP-з'єднання?», «скільки даних було передано до розриву з'єднання?», тощо. Людина-експерт використала би цю інформацію для того, щоб винести судження, або принаймні, щоб позначити потік або хост як підозріли, якщо якісь із цих та інших тверджень будуть правдиві (чи хибні). Дерева прийняття рішень здатні автоматично будувати «ланцюжки міркувань» на основі вивчених абстрактних умов, які можуть віддалено відображати послідовність людських суджень (наприклад, *якщо X та Y істинно, то, за умови Z ... а інакше...*). Тому очікувано, що цей тип класифікатора добре показав себе в ряді робіт [7][8][9] у галузі виявлення. Дерева прийняття рішень, однак, страждають від перенавчання, що є значним недоліком при виявленні втручань, оскільки навчальні набори для такого класу задач часто замалі, згенеровані штучно та містять обмежену кількість вибірок трафіку ботнетів чи атак. Це було продемонстровано в експериментальній частині цієї роботи, коли значення AUROC для класифікатора дерев прийняття рішень виявилось лише 0,477 (без використання методу головних компонентів).

Для вирішення згаданої проблеми застосовується модель «випадкового лісу» (Random Forest). Випадкові ліси мають значно меншу дисперсію, ціною нижчої точності та більшої системної похибки. Цитата Hastie et al. [10]: *«так як [випадковий ліс] є інваріантним щодо масштабування та інших перетворень значень ознак, він стійкий до включення нерелевантних ознак і утворює моделі, які легко перевіряються. Вони [випадкові ліси], однак, менш точні»*. Випадкові ліси засновані на техніці класифікації дерев прийняття рішень, але замість використання єдиного дерева, вразливого до шумів та схильного до перенавчання, в них використовується група (ансамбль) дерев. Під час навчання для кожного дерева застосовуються вибірки із підмінами, отримані з рівномірного випадкового розподілу. Для рішення задачі регресії підраховується середній результат передбачення кожного окремого дерева. У випадку задачі класифікації використовується «голосування» (вибір класу більшості). Такий підхід надає класифікатору випадкового лісу значну перевагу над деревом прийняття рішень (AUROC = 0,848 та AUROC = 0,91 при застосуванні PCA). Це особливо важливо при виявленні мережевих атак, адже є дуже мало загальнодоступних розмічених наборів даних (датаестів) для навчання, а ті, які надаються, частіше за все є недостатньо репрезентативними, що ставить проблему перенавчання моделей ще більш гостро.

Одного ізольованого потоку часто недостатньо для відтворення повної картини мережевої атаки. Тому методам однопотокової класифікації складно виявити втручання, розтягнуте в часі. Але на відміну від описаних раніше моделей, згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN) можуть враховувати контекстну інформацію.

Модель CNN виконує склейку (згортку) кількох значень ознак чи подій у єдине значення вищого рівня. Таку згортку можна зробити за значеннями однієї й тієї ж ознаки, яка змінюються в часі. Така одновимірна «склейка» називається темпоральною згорткою. Вона дозволяє описати часову мінливість ознаки з кількох пов'язаних мережевих потоків. Як було показано, це забезпечує більшу ефективність виявлення. Однак недоліком CNN моделі є те, що вона працює з групами подій фіксованого розміру. Тож,аби отримати кінцевий результат прогнозу, необхідно мати весь ланцюжок потоків, що робить їх непридатними для виявлення втручань наживо, але дуже корисними для аналізу історичних даних.

Моделі LSTM можуть приймати послідовності довільної довжини, видаючи результат класифікації для кожного нового потоку. Точність виявлення спочатку може бути низькою, але поступово підвищується, оскільки модель отримує все більше і більше історичного контексту.

Споріднені праці та порівняння ефективності моделей

Табл. 2 демонструє ефективність різних методів класифікації мережевої активності, отриманих під час роботи над цією роботою. В таблиці також наведено аналогічні показники ефективності, отримані декількома іншими дослідниками, які також використовували набір даних ISCX Botnet 2014 [4] і схожий метод виділення ознак на основі мережевих потоків. Для прямого порівняння було вибрано результати, показані моделями штучного інтелекту, подібними до застосовуваних у межах цієї статті.

Висновки

У цій праці описується повне структуроване рішення проблеми виявлення ботнетів на основі штучного інтелекту. Розглянута схема виявлення втручань не обмежена лише ботнетами і є гарним підгрунтям для майбутніх досліджень на тему виявлення будь-яких інших мережевих атак. Показники ефективності, наведені тут, можуть бути використані як еталонні величини для порівняння у майбутніх дослідженнях, а удосконалення окремих етапів схеми має забезпечити подальше покращення ефективності виявлення в цілому.

Література

1. Debar, Hervé. (2009). An Introduction to Intrusion-Detection Systems.
2. Arnaldo, Ignacio & Cuesta-Infante, Alfredo & Arun, Ankit & Lam, Mei & Bassias, Costas & Veeramachaneni, Kalyan. (2017). Learning Representations for Log Data in Cybersecurity. 250-268. 10.1007/978-3-319-60080-2_19.
3. Meshal Farhan AL-Anazi and Mostafa G M Mostafa. (2019) Efficient Botnet Detection using Feature Ranking and Hyperparameter Tuning. International Journal of Computer Applications 182(48):55-60.
4. Beigi, E.B., Jazi, H.H., Stakhanova, N., Ghorbani, A.A.: Towards effective feature selection in machine learning-based botnet detection approaches. In: 2014 IEEE Conference on Communications and Network Security. pp. 247–255 (2014)
5. Graham, Mark. (2018). A Botnet Needle in a Virtual Haystack.
6. Habibi Lashkari, Arash. (2018). CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyser for anomaly detection. URL: https://github.com/ISCX/CICFlowMeter. 10.13140/RG.2.2.13827.20003.
7. Meshal Farhan AL-Anazi, Mostafa G. M. Mostafa (2019) Efficient Botnet Detection using Feature Ranking and Hyperparameter Tuning
8. Suzan Almutairi, Saoucene Mahfoudh, Sultan Almutairi, and Jalal S. Alowibdi (2019) Hybrid Botnet Detection Based on Host and Network Analysis
9. Paulo Angelo Alves, Resende André Costa Drummond. (2018) HTTP and contact-based features for Botnet detection
10. Trevor; Tibshirani, Robert; Friedman, Jerome (2008). The Elements of Statistical Learning (2nd ed.). Springer. ISBN 0-387-95284-5.
11. Hsu, Ching-Hsiang & Huang, Chun-Ying & Chen, Kuan-Ta. (2010). Fast-Flux Bot Detection in Real Time. 6307. 464-483. 10.1007/978-3-642-15512-3_24.
12. Hwang, Chanwoong & Kim, Hyosik & Lee, Hooki & Lee, Taejin. (2020). Effective DGA-Domain Detection and Classification with TextCNN and Additional Features. Electronics. 9. 1070. 10.3390/electronics9071070.

Одержано 29.07.2022

Про авторів:

Панчук Богдан Олександрович,
аспірант відділу теорії цифрових автоматів.
Кількість публікацій в українських виданнях – 0.
Кількість зарубіжних публікацій – 0.
https://orcid.org/0000-0002-5389-359X.

Місце роботи авторів:

Інститут кібернетики ім. В.М. Глушкова НАН України,
03187, м. Київ,
проспект Академіка Глушкова, 40.
E-mail: bogdanscloud@gmail.com

Прізвища та ініціали авторів і назва доповіді англійською мовою:
B.O. Panchuk
Flow based bonet traffic detection using AI

Прізвища та ініціали авторів і назва доповіді українською мовою:
Б.О. Панчук
Виявлення бонет-трафіку на основі потоків, використовуючи Ш І

Контакти для редактора: Панчук Богдан Олександрович,
Інститут кібернетики ім. В.М. Глушкова НАН України,
e-mail: bogdanscloud@gmail.com, тел.: (38)(066) 721-74-51