ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Факултет „Компютърни системи и технологии“

Катедра „Информационни технологии в индустрията“

маг. инж. Александър Валентинов Христов

А В Т О Р Е Ф Е Р А Т

на дисертация за придобиване на образователна и научна степен „доктор”

ТЕМА: „Изследване на методи и средства на системите с изкуствен интелект с приложение в Интернет на нещата“

Област: 5. Технически науки

Професионално направление:

5.3. Комуникационна и компютърна техника

Научна специалност: 02.21.05. Системи с изкуствен интелект

Научен ръководител:

проф. д-р инж. Румен Трифонов

СОФИЯ, 2023 г.

Дисертационният труд е обсъден и насочен за защита от Катедрения съвет на катедра „Информационни технологии в индустрията“ към Факултет “Компютърни системи и технологии“ на ТУ-София на редовно заседание, проведено на 30.01.2023 г.

Публичната защита на дисертационния труд ще се състои на 16.05.2023 г. от 15:00 часа в Конферентната зала на БИЦ на Технически университет – София на открито заседание на научното жури, определено със заповед № ОЖ-5.3-26 / 21.02.2023 г. на Ректора на ТУ-София в състав:

1. Проф. дн Валери Марков Младенов – председател
2. Проф. д-р Милена Кирилова Лазарова-Мицева – научен секретар
3. Проф. д-р Александър Богданов Бекярски
4. Проф. д-р Боряна Петкова Делйска-Маноилова
5. Проф. дтн Стойчо Димитров Стойчев.

Рецензенти:

1. Проф. дтн Стойчо Димитров Стойчев
2. Проф. д-р Александър Богданов Бекярски

Материалите по защитата са на разположение на интересуващите се в канцеларията на Факултет “Компютърни системи и технологии” на ТУ-София, блок № 1, кабинет № 1443-А.

Дисертантът е докторант към катедра “Информационни технологии в индустрията” на факултет “Компютърни системи и технологии”. Изследванията по дисертационната разработка са направени от автора, като резултатите от тях са публикувани.

Автор: Александър Валентинов Христов

Заглавие: Изследване на методи и средства на системите с изкуствен интелект с приложение в Интернет на нещата

Тираж: 30 броя

Отпечатано в ИПК на Технически университет – София

I. ОБЩА ХАРАКТЕРИСТИКА НА ДИСЕРТАЦИОННИЯ ТРУД

Актуалност на проблема:

В днешно време интересът към Интернет на нещата (IoT), Индустриалният интернет на нещата (IIoT) и по-специално информационната сигурност на IoT устройствата нараства. Проблемът с информационната сигурност на IoT устройствата не е решен, нещо повече, налице са различни системи за откриване на компрометирани IoT устройства в резултат на кибератака, но те откриват само някои конкретни кибератаки. С появата на все повече IoT устройства и нови такива през последните години се появяват все повече видове атаки спрямо тях. Навлизащите нови устройства предразполагат за нови опасности, неподозирани дори и от самите производители. Нещо повече, трябва да се обърне специално внимание на защитата на IoT устройствата и тяхната правилна употреба. Всичко това определя актуалността на проблема.

Цел на дисертационния труд, основни задачи и методи за изследване

Целта на дисертационния труд е чрез конкретни методи и средства на системите с изкуствен интелект да се разработят модели за повишаване на мрежовата информационна сигурност и свързаността на устройствата в Интернет на нещата (IoT).

За постигане на целта на ДТ следва да се решат следните задачи:

1. Да се предложи метод за детектиране на компрометирани IoT устройства в резултат на кибератака, базиран на Wavelet дискретното преобразувание и Haar филтър.
2. Да се разработи система за детектиране на компрометирани IoT устройства в резултат на кибератака чрез използване на Емоционални Модели (EM).
3. Да се предложи механизъм с минимални закъснения за препращане на трафика по множество хетерогенни комуникационни канали (multipath свързаност) в IoT.
4. Да се разработи система за симулиране на кибератаки и събиране на данни за мрежовия трафик, натоварването на процесора и паметта.

Научна новост

Резултатите от дисертационния труд са в областта на сигурността и мрежовата свързаност и по- точно детектиране на компрометирани IoT устройства в резултат на кибератака чрез индексите за натоварването на процесор, памет и мрежов порт. Създадени са модели използващи Wavelet трансформации, филтър на Haar и емоционални модели на системи за детектиране на компрометирани IoT устройства и са тествани възможностите им. Предложен е подход за повишаване на качеството на обслужване при Multipath свързаност в Интернет на нещата, използващ метода на Лаграж с неопределените коефициенти и е създаден модел базиран на генетични алгоритми. Показано е увеличаването на качеството на обслужване при multipath свързаност.

Практическа приложимост

Интернет на нещата (IoT), Индустриалният интернет на нещата (IIoT) и Индустрия 4.0 / 5.0 са фактори, които се превръщат в основа на всички дейности в икономиката, администрацията, обществото и личния живот. В световен мащаб

проблемът за информационната сигурност на IoT устройствата е все още слабо разработен и липсват всеобхватни системи за детектиране на компрометирани IoT устройства в резултат на кибератака. Всяка една интелигентна система, която адресира проблема за информационната сигурност и мрежовата свързаност на IoT устройствата е с много голяма практическа приложимост, поради което получените резултати ще бъдат с национално и международно значение.

Апробация

Основните резултати от дисертационния труд са отразени в 6 научни публикации. Резултатите са дискутирани на следните международни научни конференции: International Scientific Conference “Computer Science'2022”, **IEEE Conference, Rec #55378**; International Conference on Information Technologies, **IEEE Conference, Rec # 52438**; International Scientific Conference "Engineering. Technologies. Education. Safety ‘2022. Забелязано е едно цитиране в SCOPUS.

Публикации

Основните постижения и резултати в дисертационния труд са публикувани в 6 работи, като 3 са статиии в научни списания и 3 са доклади на международни научни конференции. Две от последните 3 публикации са реферирани и индексирани в световноизвестните бази данни Scopus и Web of Science, а първите три са в списание от Националния референтен списък на съвременни български научни издания с научно рецензиране на НАЦИД.

Структура и обем на дисертационния труд

Дисертационният труд е в обем от 151 страници, като включва увод, 5 глави за решаване на формулираните основни задачи, списък с приносите, списък на публикациите по дисертацията и използвана литература. Цитирани са общо 221 информационни ресурси, като 156 са литературни източници, от тях 12 - на кирилица, а останалите - на латиница. Работата включва общо 47 фигури и 17 таблици. Номерата на формулите, фигурите и таблиците в автореферата съответстват на тези в дисертационния труд.

II. СЪДЪРЖАНИЕ НА ДИСЕРТАЦИОННИЯ ТРУД

Глава 1. Аналитичен обзор на IoT, МИС и ИИ

В тази глава е направен обширен аналитичен обзор на съвременното състояние на проблема, като в резултат може да се направят следните изводи:

•Известни са множество подходи за идентифициране на компрометирани IoT устройствата, особено в рояци от сензори за IoT – например, съхраняват се поведенчески файлове и периодично устройството трябва да генерира файлове, които трябва да бъдат същите като еталоните. Използват се предимно хеширане и криптография или удостоверяване на публичен ключ. NICE включва всички процеси, необходими, за да се гарантира, че съществуващите и новите ИТ системи отговарят на изискванията за киберсигурност и риск на организацията. В NICE Protect and Defend се дискутира извършването на оценки на заплахи и уязвимости; определяне на отклонения от приемливи конфигурации или политики; оценка на нивото на риск; и разработване или препоръчване на подходящи смекчаващи мерки.

Всичките тези дискутирани и анализирани по-рано в тази глава решения имат своите предимства и недостатъци и оттам ограничено приложение.

Оттук следва необходимостта да се разработят модели с приложение за повишаване на мрежовата информационна сигурност и свързаността на устройствата в IoT.

Глава 2. Метод за откриване на компрометирани IoT устройства в резултат на кибератака

В тази глава е разгледано съвременното състояние на идентифицирането на компрометирани устройства в резултат на реализирана кибератака чрез мониторинг на използването на процесорните ядра, паметта и мрежовите интерфейси на IoT устройствата. Възможно е разработване на теоретичен метод, който да използва weak изкуствен интелект и по-точно Wavelet трансформация за детектиране на компрометирани IoT устройства. Wavelet трансформацията [П1] е трансформация, която осигурява едновременно представяне на сигнала във времева и честотна област. Тя предава сигнал във времевата област през високочестотен и нискочестотен филтри, като филтрира съответно ниската и високата честота на сигнала. Процедурата се повтаря, докато част от сигнала, съответстващ на дадена честота се премахва. Процедурата се нарича разлагане. Разлагането се повтаря до предварително зададено ниво на декомпозиция. След това се образува множество от сигнали, които всъщност представят оригиналния сигнал.

Върху индексите за натоварването на паметта на IoT устройството, тук се използва Wavelet дискретното преобразуване. Haar трансформацията разлага дискретния сигнал на два подсигнала. Единият подсигнал е текущата средна стойност или тенденция – T, а другият подсигнал е текущата разлика или флуктуация - d.

Целта на настоящата глава е на базата на анализ на системите за информационна сигурност да се разработи модел за идентифициране на компрометирани IoT устройства в резултат на кибератака, да се предложи алгоритъм и система за идентифициране на компрометирани IoT устройства използваща Wavelet трансформации и филтър на Haar и да се уточнят параметрите на модела, така че да се детектират най-точно двете състояния на IoT устройствата - нормално работещи или компрометирани. Предложени са два варианта.

Вариант 1

Известно е, че най-голямо значение за качественото идентифициране на състоянията на IoT устройствата имат времевите редове за натоварването на паметта. Предложеният тук метод за идентифициране на различни компрометирани състояния на IoT устройства използва като мярка енергията, получена чрез Wavelet трансформацията върху натоварването на паметта на IoT устройството. Методът за тестване се състои от две фази. В началната фаза (Първоначална работа в продуктивна среда) когато IoT устройството е некомпрометирано, Wavelet енергията се измерва и запаметява като еталонна такава:

(2.1) $$\bar{E}_T = \frac{1}{n}\sum_{i=1}^{n} E_{T,i},$$ където

$\bar{E}_T$ - средната стойност на Wavelet енергията на некомпрометирани състояния;

$E_{T,i}$ - енергията на натоварването на паметта на IoT устройството при n-те на брой различни некомпрометирани състояния.

Във втората фаза (Тестова фаза), Wavelet енергията на тестваното IoT устройство се измерва и сравнява със съответната еталонна стойност:

(2.2) $\Delta = E_{T,j} - \bar{E}_T$, където

$E_{T,j}$ – енергията на натоварването на паметта на IoT устройството при тестване на състоянията му;

$\bar{E}_T$ - средната стойност на Wavelet енергията на некомпрометирани състояния.

Състоянията се определят като компрометирани, когато Wavelet енергията превишава зададен праг на толеранс:

(2.2) $E_{thresh} = \theta \times \bar{E}_T$, където

θ – коефициент за праг на толеранс;

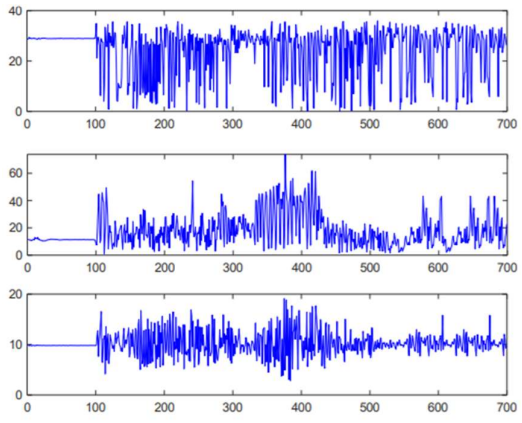
$\bar{E}_T$ - средната стойност на Wavelet енергията на некомпрометирани състояния.

Прагът на толеранс е въведен, за да се вземат под внимание вариациите на натоварването на паметта на IoT устройството при различни компрометирани състояния и неточностите от измерването (системния монитор). Накрая на всеки цикъл (итеративно) се преизчисляват $\bar{E}_T$ за Wavelet енергията на некомпрометираното IoT устройство и границата на толеранс на откриваемост на компрометираните състояния - E_{thres} .

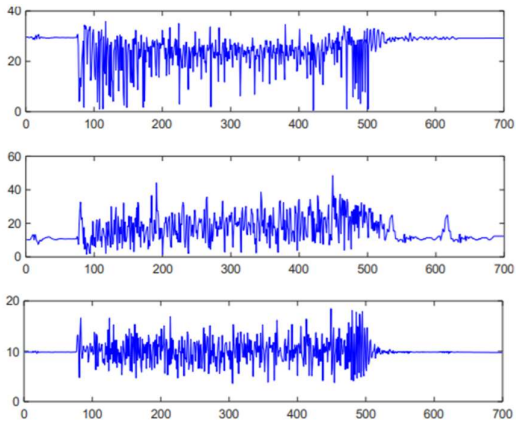
Трябва да се отбележи, че стойността на коефициента θ оказва влияние върху откриваемостта на компрометирани състояния и следва да се избере евристично, на базата на предишен опит (Вж. по-горе „Първоначална работа в продуктивна среда“) и/или от литературни източници.

Експеримент

С цел да се сравнят експерименталните резултати със съществуващи подобни реализации на системи за идентифициране на различни компрометирани състояния на IoT устройства тук са използвани изходните данни от публикация [125] от 2020г. за натоварването на паметта на IoT устройство. На **фиг. 2.2** и **2.3** съответно са показани графично времевите редове за индексите за натоварването на процесор, памет и мрежов комуникационен интерфейс за некомпрометирано и компрометирано IoT устройство. Извършена е обработка на изображенията чрез авторска приложна програма написана на C. В github хранилище на автора [ПР1] са дадени съответно предложената програма на C и получените bmp и csv файлове (данните са представени графично и таблично).



фиг. 2.2



фиг. 2.3

На **фиг. 2.2** най-горе е графиката за натоварването на паметта на IoT устройството, в случаите, когато същото е некомпрометирано, а на **фиг. 2.3** най-горе е графиката за натоварването на паметта на IoT устройството, в случаите, когато същото е компрометирано. Вторите графики са за натоварването на процесора, а третите – за използването на мрежовия комуникационин интерфейс.

Разработен е авторски скрипт на Python [ПР2].В скрипта чрез последователното извикване на библиотечната функция ***pywt.dwt (x, 'haar')*** итеративно се пресмятат коефициентите на тенденцията (за първото, второто, ..., n-тото ниво на разлагане) до получаването на един елемент в масива T от тип ***numpy.ndarray***, отчитайки само коефициентите на тенденцията на предишното ниво на разлагане. Последно полученият коефициент (подсигналът на тенденцията) T се повдига на квадрат за да се пресметне енергията $E_{T,j}$. Основната част на скрипта е показана **фиг 2.6**.

```
cwd = os.getcwd()
df = pd.read_csv(cwd + "/CSV/mem1.csv")
testValues = df['Value'].to_list()
T , D = pywt.dwt(testValues, 'haar')
while len(T) > 1:
    testValues = T
    T , D = pywt.dwt(testValues, 'haar')
```

фиг. 2.6 Python скрипт за изчисляване на подсигналът на тенденцията T

Резултатите от работата на Python скрипта върху данните за натоварването на паметта на IoT устройството, в случаите, когато е некомпрометирано, и когато е компрометирано са показани в *табл. 2.1*. В първия ред са дадени коефициентът на тенденцията - T и енергията - E за натоварването на паметта на некомпрометираното IoT устройство, а във втория ред - коефициентът на тенденцията - T и енергията - E за компрометираното IoT устройство. Таблицата е допълнена с трети ред, в който е дадена разликата в проценти между енергията E за компрометираното и некомпрометираното състояние на IoT устройство.

табл. 2.1 Резултати за натоварването на паметта на IoT устройство

IoT устройство	T	E
Некомпрометирано	539.2	290726
Компрометирано	567.3	321829
Разлика в енергията		10.7 %

Аналогично са получени резултати за разликата в проценти между енергията E за компрометираното и некомпрометираното състояние на IoT устройство от времевите редове за натоварването на процесора и мрежовия интерфейс на не/компрометирано IoT устройство, които съответно са показани в *табл. 2.2* и *табл. 2.3*.

табл. 2.2 Резултати за натоварването на процесора на IoT устройство

IoT устройство	T	E
Некомпрометирано	420.3	176652
Компрометирано	430.5	185330
Разлика в енергията		4.91%

табл. 2.3 Резултати за натоварването на мрежовия интерфейс на IoT устройство

IoT устройство	T	E
Некомпрометирано	231.8	53731
Компрометирано	227.2	51620
Разлика в енергията		-3.93 %

От сравняването на получените резултати използвайки индексите за натоварването на паметта (*табл. 2.1*), процесора (*табл. 2.2*) и мрежовия интерфейс (*табл. 2.3*) се вижда, че най- голяма е разликата между енергията E за компрометираното и некомпрометираното състояние на устройството при натоварването на паметта ΔE =10.7 %. Това напълно съответства на резултатите и изводите от [125], което е вид верификация на получените резултати. От друга страна, целесъобразно е стойността на θ да се избере равна на 0.1 (вж. формула 2.2), т.е. малко под ΔE =10.7 % (*табл. 2.1*), за да се вземат под внимание вариациите на натоварването на паметта на IoT устройството при различни компрометирани състояния и неточностите при мониторинга.

Вариант 2

Както е известно в редица случаи IoT устройствата са с неголяма изчислителна мощ. Ето защо е целесъобразно да се минимизира изчислителната сложност на предложения по-горе алгоритъм.

Във втория вариант на алгоритъма, енергията на преобразувания сигнал - E_T се пресмята само от коефициентите на енергията на тенденцията - T, и по- точно, енергията E_T се пресмята от коефициентите на тенденцията на първото ниво на разлагането. Разработен е авторски скрипт на Python, в който се извиква библиотечната функция ***pywt.dwt (x, 'haar')***, след което се пресмята енергията E (сумата от квадратите на коефициентите на тенденцията). Основната част на скрипта е показана **фиг 2.7**.

```
cwd = os.getcwd()
df = pd.read_csv(cwd + "/EMO-Calculated/Square/Pr3.csv")
testValues = df['Value'].to_list()
cA , cD = pywt.dwt(testValues, 'haar')
sum = 0

for i in range(0, len(cA)):
    sum = sum + pow((cA[i]*100),2)
```

фиг. 2.7 Python скрипт за изчисляване на енергията E

Експеримент

Резултатите от работата на Python скрипта върху данните за натоварването на паметта на IoT устройството, в случаите, когато е некомпрометирано, и когато е компрометирано са показани в *табл. 2.4*. В първия ред са дадени енергията - E за натоварването на паметта на некомпрометираното IoT устройство, а във втория ред - енергията за компрометираното IoT устройство. Таблицата е допълнена с трети ред, в който е дадена разликата в проценти между енергията E за компрометираното и некомпрометираното състояние на IoT устройство.

табл. 2.4 Резултати за натоварването на паметта на IoT устройство

IoT устройство	E
Некомпрометирано	315904,4
Компрометирано	343753,5
Разлика в енергията	8,8 %

Аналогично са получени резултати за натоварването на процесора и мрежовия интерфейс на не/компрометирано IoT устройство. Получените резултати за натоварването на процесора и мрежовия интерфейс са показани съответно в *табл. 2.5* и *табл. 2.6*.

табл. 2.5 Резултати за натоварването на процесора на IoT устройство

IoT устройство	E
Некомпрометирано	210209.9
Компрометирано	203995.6
Разлика в енергията	-3,0 %

табл. 2.6 Резултати за натоварване на мрежовия порт на IoT устройство

IoT устройство	E
Некомпрометирано	55775.2
Компрометирано	53406.4
Разлика в енергията	4,2 %

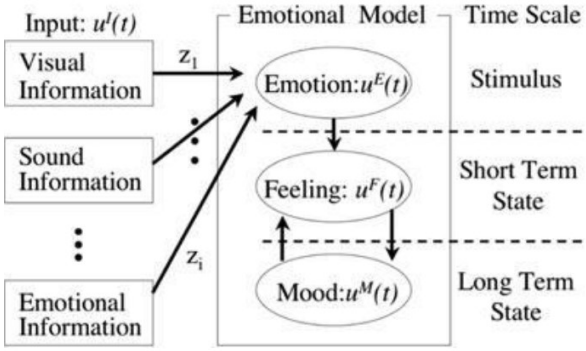
И тук индексите за натоварването на паметта са от най-голямо значение за успешното идентифициране на компрометираното състояние на IoT устройството. Както е видно от *табл. 2.1* и *табл. 2.4* разликите в енергиите за натоварването на паметта на компрометирано и некомпрометирано състояние при олекотения алгоритъм са по-малки (8,8%<10.7%) и ето защо по-целесъобразно е в интелигентната система за детектиране на компрометирани IoT устройства да се използва първия вариант на алгоритъма.

Глава 3. Система за откриване на компрометирани IoT устройства в резултат на кибератака чрез използване на Емоционални Модели

В гл. 1 на ДТ са дискутирани особеностите при обработката на времеви редове (класически модели и невронни мрежи). За съжаление, предложения в гл. 2 метод за детектиране на компрометирани IoT устройства в резултат на различни кибератаки, включително и нови такива дава разлика малко над 10%, което е от порядъка на грешката в инженерната практика. Невронните мрежи дават много добра точност при разпознаването на известни кибератаки, но не са приложими за новопоявяващите се такива. Остава отворен въпросът за детектиране на компрометирани IoT устройства в резултат на кибератака, като се използват индексите за натоварването на процесор, памет и мрежовия интерфейс.

Целта на настоящата глава е да се предложи система за детектиране на компрометирани IoT устройства в резултат на кибератака на базата на емоционални модели (ЕМ), като се използват индексите за натоварването на процесор, памет и мрежовия интерфейс, както и да се уточнят параметрите на ЕМ, така че максимално лесно да се разграничават двете състояния на IoT устройствата - нормално работещи или компрометирани.

Емоционалния Модел [П3] е такъв, в който се прилагат концепциите за емоция, чувство и настроение въз основа на времева скала, приемайки че емоциите се променят във времето въз основа на възприеманата информация за вътрешното състояние и външната среда.



фиг. 3.1 Емоционален модел

Концепцията за емоционален модел (**фиг. 3.1**) включва: емоция, чувство и настроение.

Емоцията се смята за интензивно краткотрайно ментално състояние, базирано на възприетата информация и използвано като междинен вход от възприятелната система към емоционалния модел. Емоцията зависи от възприятелната информация и всяко чувство се актуализира на базата на емоциите. Емоциите $u_{j,i}^E$ се определят от конкретните възприятия/входове (j) на възприятелната система по следния начин:

(3.1) $$u_{j,i}^E(t) = z_{j,i} u_j^I(t),$$ където

$z_{j,i}$ е коефициента на допринасяне на възприятията от вход (j), към емоцията (i)

Чувството се дефинира като краткосрочно състояние, актуализирано от промяна в емоцията. Чувствата са по-дълготрайни и по-устойчиви от емоциите. Погледнато отгоре-надолу на **фиг. 3.1**, състоянието на i-тото чувство $u_i^F(t)$ се определя/актуализира от конкретните влияещи му емоции отгоре и от настроението $u^M(t)$ отдолу:

(3.2) $$u_i^F(t) = \kappa u_i^F(t-1) + (1 - \kappa) \sum_{j=1}^{N'} u_{j,i}^E(t)$$

(3.3) $$\kappa = \frac{\gamma^F}{1 + u^M(t-1)},$$ където

γ^F е коефициент на затихване на чувствата ($0 < \gamma^F < 1$).

Настроението (Mood) се определя като дългосрочно състояние, актуализирано от промяната на чувствата, и управлява промените в чувствата. Текущата стойност за настроението се актуализира от сумата от чувства:

(3.4) $$u^M(t) = \gamma^M u^M(t-1) + \frac{1-\gamma^M}{N^F} \sum_{i=1}^{N^F} u_i^F(t),$$ където

γ^M е коефициент на затихване на настроението ($0 < \gamma^M < 1$) и N^F е броят на чувствата.

Експеримент

Използват се данните от гл. 2 за некомпрометирано/компрометирано IoT устройство за използването на паметта, процесора и мрежовия интерфейс. Тези данни са възприятията (данните за входовете- $u_j^I(t)$, j=1,2,3) към емоциите.

Броят на емоциите е избран да е 3 - E1, E2 и E3, като първата емоция, E1 е с преобладаващо тегло за натоварването на процесора; втората, E2 - с използването на мрежовия интерфейс и трета, E3 - с използването на паметта.

Броят на чувствата също е избран 3. По този начин допълнителен приоритет може да бъде присвоен на всяко чувство при уточняване на модела. Стойностите на параметрите - степента на принос $z_{j,i}$, γ^F и γ^M следва да са избрани такива, че разликата в настроението за компрометирано и некомпрометирано IoT устройството е максимална.

Първите числени резултати са получени с помощта на Web-базиран калкулатор на емоционални модели, на който се подава XML (eXtensible Markup Language) файл. Входът за калкулатора на емоционалния модел е XML файла с предварително дефинирана структура. Този XML файл описва и емоционалния модел и необходимите му данни. Съдържанието на XML файла е показано в *таблица 3.1*.

табл. 3.1 ЕМ за идентифициране състоянията на IoT устройства

01	Time	mem	nic	pr	F1	F2	F3	Mood
02	0.85	e	e	e	f	f	f	m
03	0.9	#FF9933	#FF8000	#FF0080	#CC001A	#99330	#FF0000	#000000
04	F1/AFL	0.05	0.05	0.9	1	1	1	1
05	F2	0.05	0.1	0.85	0	0	0	0
06	F3	0.05	0.15	0.8	0	0	0	0
07	RSRV.	0	0	0	0	0	0	0
08	0	0.291071	0.099107	0.134375	‘*Row 01 – the first column is named “Time” and other columns are named on observed emotions, feelings and mood;* *Row 02 – in the first column is given the discount rate of the feelings, in the next columns are given the markup symbols for the emotions “e”, for the feelings “f” and for the mood “m”;* *Row 03 – in the first column is given the discount rate of the mood and in the next columns are specified the colors for the graphical presentation of each emotion, feeling and mood;* *Row 04 –... (in this case 04 – 06) in the first column are repeated the names of the feelings and in the next columns for the emotions (marked with “e”) is given the corresponding participation of emotions in the definition of feelings. Note, the last four columns (titled as AFL-Alert Filter Level gives the minimum values over which the system generates alert) are not used here;* *Rows after that (numbered 0...n in the column Time) include the values of monitored emotions in every one observed moment in time between 0 and n.*’[132]					
09	1	0.291071	0.100893	0.13125				
10	2	0.291071	0.100893	0.13125				
...	...	...	...	...				
n	524	0.2625	0.111607	0.221875				

Уеб калкулаторът на емоционални модели чете данните и правилата на емоционалния модел (Вж. формули (3.1)-(3.3)), изчислява стойностите на чувствата и настроението и визуализира резултата.

Модификация на уеб калкулатора и съответните XML документи са дадени в [ПР3] и по-точно XML файловете с коефициентите и данните за натоварването на

паметта, мрежовия интерфейс и процесора на IoT устройството, в случаите, когато е некомпрометирано (EMOCOL_Z1.xml) и когато е компрометирано (EMOCOL_Z2.xml).

Интерес представляват последните стойности за настроението от калкулатора на емоционалните модели (n=524 в колоната Time) за компрометирано и некомпрометирано състояние на IoT устройството и по-точно разликата между тези две стойности. Същите са показани в *табл. 3.2*. Първият ред на таблицата съдържа настроението за некомпрометираното IoT устройство, вторият ред - настроението за компрометираното устройство, а третият ред на таблицата показва разликата по абсолютна стойност, в проценти, между настроението за двете състояния (компрометирано и некомпрометирано) на IoT устройството.

табл. 3.2 Резултати за състоянията на IoT устройствата

IoT устройство	Mood
Non-compromised	0.310
Compromised	0.274
Δ	12.4 %

Експерименталните резултати и по-точно индексите за настроението (*табл 3.2*) показват, че разликата между настроението на компрометираното и некомпрометираното IoT устройство е Δ = 12,4%. Това съответства на резултатите и изводите от [125] и също така е верификация за получените резултати.

По-долу се дискутира уточняването на параметрите или оптимизирането на ЕМ посредством намирането на такива $z_{i,j}$, γ^F и γ^M, че разликата между настроението на компрометираното и некомпрометираното IoT устройство да е максимална.

Оптимизиране на емоционалния модел

Предложеният подход, базиран на емоционални модели за откриване на компрометирани IoT устройства при кибератака, включително новопоявила се такава е обещаващ, но е необходимо провеждането на пълен факторен експеримент за намиране на такива коефициенти $z_{i,j}$, γ^F и γ^M на модела, че разграничаването на компрометирано от некомпрометирано устройство да бъде максимално, т. е. Δ -> max

Броят на коефициентите $z_{i,j}$ при 3 емоции и 3 чувства, γ^F и γ^M (вж. *табл. 3.1*) е 3x3+2=11 и дори при 10 нива на същите е необходимо провеждането на 10^{11} експеримента, т.е.ще са нужни $2x10^{11}$ пускания на уеб калкулатора на емоционални модели. Целесъобразно е да бъде разработен калкулатор на емоционални модели в случая – на програмния език Python, така че да се автоматизира работата по провеждането на пълен факторен експеримент за намиране на оптималните коефициентите $z_{i,j}$, γ^F и γ^M в модела.

Така разработеният калкулатор на емоционални модели е даден в [ПР3] се стартира в Jupyter Notebook среда. С този калкулатор първо ще се проведе пълен факторен експеримент за коефициентите $z_{i,j}$ при конкретни коефициенти на затихване на чувствата и настроението - γ^F и γ^M. При така намерените оптимални $z_{i,j}$ ще се проведе и втори експеримент за оптимизиране на коефициентите на затихване на чувствата и настроението - γ^F и γ^M. Пълният код на скрипта за намиране на оптималните коефициенти $z_{i,j}$ е даден в приложение [ПР2].

Стъпките на алгоритъма са описани по-долу:

Стъпка 0: Дефинира се метод ***calculateMood***, който връща изчисената стойност за настроението, по формули (3.1) – (3.4) от данните за натоварването на паметта, процесора и мрежовия интерфейс от подавания като формален параметър csv файл. Също така се дефинира метод ***saveToList***, който записва данните от изчисленията в списъци, които са създадени в началото на програмата;

Стъпка 1: Импортират се нужните библиотеки и се инициализират списъци, които ще се използват при работата на програмата;

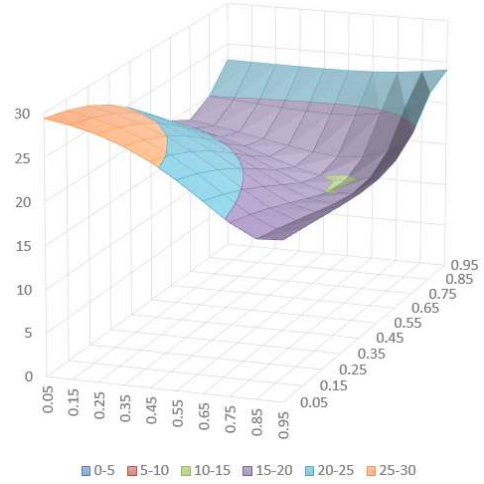
Стъпка 2: Задават се стойностите на коефициентите γ^F и γ^M;

Стъпка 3: Във вложени цикли се изчисляват стойностите за настроението за компрометирано и некомпрометирано състояние, като на всяка итерация съответният коефициент $z_{i,j}$ се увеличава с 0.1. След това се изчислява разликата между стойностите за настроението за компрометирано и некомпрометирано състояние и се записва в променливата ***delta***;

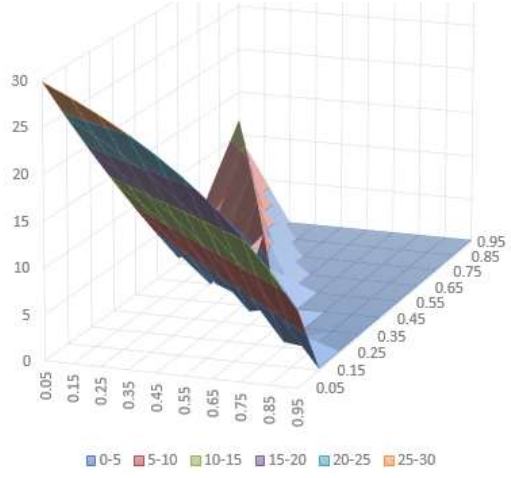
Стъпка 4: Записват се получените резултати в csv файл, като за целта списъците се обединяват в dictionary ***dict*** и същите се конвертират в DataFrame ***df***.

Максимална разлика между настроенията за компрометирано и некомпрометирано устройство - Δ (в %) се получава при конкретни "оптимални" стойности на коефициентите $z_{i,j}$. Както бе дискутирано по-рано, проведен е и пълен факторен експеримент за коефициентите на затихване на чувствата и настроението- γ^F и γ^M при конкретни $z_{i,j}$ На **фиг. 3.8** е дадена графика, показваща разликата (делта), в % между настроенията за компрометирано и некомпрометирано устройство за различните γ^F и γ^M (коефициентите на затихване на чувствата и настроението) при следните $z_{i,j}$: z11 = 0.05, z12 = 0.1, z13 = 0.85, z21 = 0.05, z22 = 0.1, z23 = 0.85, z31 = 0.05, z32 = 0.1 и z33 = 0.85.

Видно от дадените в приложението резултати е, че разликата Δ е максимална при стойности на коефициентите $z_{i,1}$=0.05, $z_{i,2}$=0.05 и $z_{i,3}$=0.9 ($i = \overline{1,3}$). Зависимостта на разликата на настроението за компрометирано и некомпрометирано състояние от стойностите на $z_{i,j}$ е показана на **фиг. 3.11**. Формата на повърхнината е такава, тъй като изобразената разлика Δ е по абсолютна стойност (Вж. (3.5)).



фиг. 3.8



фиг. 3.11

Настроението (Mood) се определя като дългосрочно състояние, актуализирано от промяната на чувствата, които пък се определят/актуализират от конкретните влияещи му емоции, т.е. емоционалният модел реализира натрупвания във времето на емоциите (индексите mem, nic и pr). Обяснението за оптималните стойности на коефициентите $z_{i,j}$ (за процесора- $z_{i,3}$=0.9, а за паметта- $z_{i,1}$, и за nic- $z_{i,2}$ са 0.05, т.е. в пъти по-малки) е, че разликата в девиацията за натоварването на процесора на некомпрометираното и компрометираното устройство е 0,14 докато тази разлика за паметта и мрежовия интерфейс е в пъти по- малка. Вж. стойностите за девиацията в табл. 2.7.

Глава 4. Механизъм за препращане на трафика при Multipath свързаност в Интернет на нещата

Multipath свързаността е характерна както за сензорните мрежи, така и за гръбнака на IoT. От гледна точка броя на устройствата в Интернет на нещата и обема данни, който тези устройства, създават съществуват два метода за формиране на IoT свързаност. При първия граничните или крайните сензори и устройства директно се свързват в облака. При втория - сензорите на граничното ниво образуват групи и клъстери около шлюзовете и маршрутизаторите за осигуряване на междинни области, преобразуващи протоколи, управляващи и реализиращи обработката на границата (fog processing). Първият модел се явява по-сложен и по-скъп (по-високопроизводителни сензори, преобразуватели, пограничи устройства) за разлика от втория модел, при които е по-сложно осигуряването на мрежова сигурност (например, Internet Protocol Security- IPSec VPN, Generic Routing Encapsulation - GRE).

Свързаността в Интернет на нещата се реализира посредством:

- безжични персонални мрежи (WPAN), които не са базирани на IP (например, Bluetooth 5, Zigbee и Z-Wave протоколи, и сензорни мрежи на основата на клетъчна топология);
- WPAN и WLAN на основата на IP (включително стандартите 6LoWPAN, Thread, IEEE 802.11p за транспортни системи и 802.11ah за Интернета на нещата);
- Системи и протоколи за далечна свързаност позволяващи обмен на данни между Интернет на нещата и облака (включително стандартите за клетъчни мрежи 5G, 6G и технологиите LoRaWAN, Sigfox, и др).

Всички тези интерфейси и протоколи реализиращи свързаността при IoT се явяват хетерогенни (с различни скорости) и позволяват да се организират multipath канали (множествени връзки) от точка до точка.

Устройствата в Интернет на нещата с функции в реално време изискват взаимодействие между кибер и физическия свят. Особеност на Интернет на нещата в реално време (Real Time-IoT) е, че има строги изисквания за реализираните времезакъснения. Всякакви проблеми, които пречат на нормалната работа на устрйствата, могат да доведат до срив в системата или да представляват заплаха за безопасността на хората. Ето защо функциите за управление на времезакъсненията, скоростта на трафика, респективно качеството на услугите- Quality of Services (QoS) са от много голямо значение при IoT, когато мрежата работи в режим с променливо натоварване и/или претоварване.

Множеството пътища (Multipath технологиите) увеличават пропускателната способност но за да се подобри качеството на услугата - QoS е необходимо да се минимизира времето за предаване на пакети чрез тези няколко пътя.

По-долу се предлага подход за оптимизиране на времето за чакане на пакети при Multipath технологии като се анализират съответния механизъм за обмен на

пакетите и конкретни протоколи използвани при Multipath технологии като MPT-GRE, MPTCP и др.

В настоящата глава се предлага подход за оптимизиране на предаването на пакети по множество мрежови канали. Подобряването на свързаността на IoT, респективно QoS е свързано с намаляване на времезакъсненията. Това е от много голямо значение за управление в реално време при IIoT, мрежите за видеонаблюдение, преноса на звук и други сигнали в IoT.

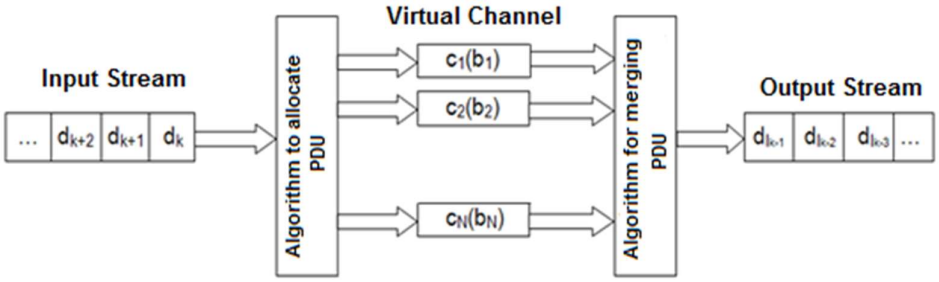
Подход за оптимизиране на предаването на пакети с множество пътища [П4]

Multipath технологиите създават множество пътища (виртуални канали - VC) за всяка двойка възли за доставяне на пакети до дестинацията.

Тук е предложен подход за оптимизиране на предаването на пакети по множество пътища за доставяне на трафик между две комуникационни точки (хостове) и динамично наблюдение и адаптиране на натоварването на трафика, разпределено към всеки виртуален канал. Подходът не трябва да изисква интензивни изчисления, поради ограниченията на производителността на процесора на хостовете.

На **фиг. 4.3** е показано разпределянето на блоковете данни към множеството канали. Блоковете данни на протокола (Protocol Data Unit - PDU) са опаковани последователно в пакети и кадри. На входа на системата постъпват данни D={dk}, като k ϵ (1, 2,…, ∞) с интензивност λ. Всеки PDU има дължина s_k, k ϵ (1, 2... ∞). Средният размер на блока данни на протокола е означен с s.

Както може да се види на **фиг. 4.3**, системата управлява N канала C=(c_1, c_2,…c_N), образуващи виртуален канал за данни. За всеки канал c_i ϵ C, i ϵ {1,2,.. N} е зададена максималната широчина на честотната му лента b_i, в bit/s, респективно скоростта му $\mu_i=b_i/s$, измервана в пакети за единица време, $pkts^{-1}$.



фиг. 4.3 Прехвърляне на пакети

Разпределящият алгоритъм (**фиг. 4.3**) разпределя данните към един от каналите, който е на разположение: ∀d_k, k ϵ (1,2 ,..., ∞), e_i:c_i ϵ C. Алгоритъмът за обединяване слива получените блокове данни от комуникационните канали в един поток, като блоковете се пренареждат така, че да са в правилен ред l_1, l_2, …, l_{k-1}, l_k.

Разпределящият алгоритъм разпределя данните към всеки от каналите, с вероятност $p_i=\lambda_i/\lambda$, i ϵ {1,2,.. N}, такава, че да се минимизира времето за предаване на пакетите, т. е.:

(4.1) $$\sum\lambda_i t_i \text{ -> min}$$

След като се намерят оптималните вероятности p_i, разпределящият алгоритъм разпределя данните към даден канал, като тегли случайно число в интервала (0,1) и определя в кой от дадените интервали:
{(0, p_1), (p_1, p_1+ p_2),… , (p_1+ p_2+...,1)} попада това число.

Предлаганият в настоящата секция подход за разпределяне на трафика цели да се минимизира времето за чакане на пакетите в маршрутизаторите при хетерогенни комуникационни канали. За да се реши тази задача по-долу се използват известните от математиката метод на Лагранж с неопределените коефициенти и формула за времето за чакане в M/M/1 система за масово обслужване:

(4.2) $$t_i = \frac{1}{\mu_i - \lambda_i}$$

където μ_ι - скоростта на предаване на пакетите и λ_ι - интензивността на пристигане на пакетите в канал c_i , i ϵ {1,2,.. N} и $\mu_\iota > \lambda_\iota$.

Съгласно метода на Лагранж:

(4.3) $$\begin{cases} \frac{\partial \sum_{i=1}^{N} \lambda_i \, t_i}{\partial \lambda_1} + \lambda\xi = 0 \\ \frac{\partial \sum_{i=1}^{N} \lambda_i \, t_i}{\partial \lambda_2} + \lambda\xi = 0 \\ \dots \\ \frac{\partial \sum_{i=1}^{N} \lambda_i \, t_i}{\partial \lambda_i} + \lambda\xi = 0 \\ \dots \\ \sum_{i=1}^{N} \lambda_i = \lambda \ , i \in \{1,2,..N\} \end{cases}$$

След заместване на t_i от (4.2) в (4.3), за λ_i се получава:

(4.5) $$\mu_i + \lambda\xi\left(\mu_i - \lambda_i\right)^2 = 0$$

За да има решение уравнението (4.5), то ξ е отрицателно, понеже λ_i и μ_i, i ϵ {1,2,.. N} са скорости, т.е. са положителни числа.

Уравнението (4.5) има единствен корен в областта на допустими стойности, понеже $\mu_\iota > \lambda_\iota$:

(4.6) $$\lambda_i = \mu_i - \sqrt{\frac{\mu_i}{-\lambda\xi}}$$

Заместваики (4.6) в последното уравнение от системата уравнения (4.3), за члена, в който участва ξ се получава:

(4.7) $$\sqrt{\frac{1}{-\lambda\xi}} = \frac{\sum_{j=1}^{N} \mu_j - \lambda}{\sum_{j=1}^{N} \sqrt{\mu_j}}$$

Заместваики в (4.6) с (4.7) получаваме търсената интензивност- λ_ι:

(4.8)
$$\lambda_i = \mu_i - \left(\sum_{j=1}^{N} \mu_j - \lambda \right) \frac{\sqrt{\mu_i}}{\sum_{j=1}^{N} \sqrt{\mu_j}}$$

Накрая, като се вземе предвид, че $\mu_i = b_i/s$, за интензивностите, с които алгоритъма разпределя данните към всеки от каналите c_i, λ_ι i ϵ {1,2,.. N} се получава:

(4.9)
$$\lambda_i = \frac{b_i}{s} - \left(\sum_{j=1}^{N} \frac{b_j}{s} - \lambda \right) \frac{\sqrt{b_i}}{\sum_{j=1}^{N} \sqrt{b_j}}$$

В случай, че $\lambda_\iota < 0$, тогава трябва да се приравни на 0 и да се реши горната задача с ограничението $\lambda_\iota = 0$.

Респективно за вероятността p_i се получава:

(4.10)
$$p_i = \frac{b_i}{\lambda s} + \left(1 - \frac{1}{\lambda s} \sum_{j=1}^{N} b_j \right) \frac{\sqrt{b_i}}{\sum_{j=1}^{N} \sqrt{b_j}}$$

Предложеният механизъм разпределя PDU към i-тия VC с вероятност p_i. Накрая, алгоритъмът за разпределяне, използващ тези оптимални вероятности p_i, i ϵ {1, 2… N}, разпределя PDU на всеки VC чрез алгоритъма RoundRobin.

При протокола MPT (Multi-Path Tunnel) пакетите се разпределят към всеки един от каналите така, че натовареността им да е еднаква, т. е.

(4.11)
$$\rho_1 = \rho_2 = \cdots = \rho_i = \cdots = \rho_j = \frac{\lambda_i}{\mu_i} = \frac{\lambda_j}{\mu_j} = \frac{\lambda}{\mu}$$

където μ_ι - скоростта на предаване на пакетите и λ_ι - интензивността на пристигане на пакетите в канал c_i , i ϵ {1,2,.. N}

Респективно:

(4.12)
$$\lambda_i = \frac{\mu_i}{\mu} \lambda$$

Откъдето вероятността p_i, с която пакетите се разпределят към всеки един канал е:

(4.13)
$$p_i = \frac{\lambda_i}{\lambda} = \frac{\mu_i}{\mu}$$

Сравнителен анализ на механизмите MPT и предложения OPTM

По-долу се изследват закъсненията за механизмите MPT (Multi-Path Tunnel) и предложения тук OPTM (Optimized Packet Transmission Mechanism). Изследването е проведено при следните входни параметри:

- среден размер на пакетите данни - s= 1500b;
- брой на каналите - N=2;
- скоростта на първия канал е четири пъти по-малка от вторият канал (b_1=300Mbps и b_2=1200Mbps);
- интензивността на постъпване на пакетите се изменя от 0,1 до 0,9 от общата скорост на каналите (1500Mbps).

При MPT пакетите се разпределят към всеки един от каналите (с вероятности p_1=0.2 и p_2=0.8) така, че натовареността на каналите да е еднаква, т.е. в случая алгоритъмът ще разпределя към втория канал четири пъти повече пакети от колкото към първия канал.

При OPTM интензивността на пристигане на пакети също е 150, 450, 750, 1050 или 1350 Mbps, т.е. е 0,1, 0,3, 0,5, 0,7 и 0,9 от общата честотна лента на VC (b_1+ b_2=1500Mbps), и за всяка една от тези стойности {150, 450, 750, 1050 и 1350 Mbps} са изчислени оптималните интензивности/ вероятности, с които механизмът препраща данни към двата канала. Резултатите са показани в *табл. 4.1*.

табл 4.1 Първи резултати за оптимални интензивности по (4.8)

λ	150	450	750	1050	1350
λ_1	-150	-50	50	150	250
λ_2	300	500	700	900	1100

Не е трудно да се провери, че по (4.9) при малки натоварвания (λ<0.4 μ), в случая λ = 150 и 450 Mbps, за λ_2 се получават отрицателни стойности. В тези случаи авторът предлага съответните интензивности да се изберат равни на нула (λ_2=0) и да се повтори решаването на оптимизационната задача. Така получените резултати за λ_1 и λ_2 са дадени в *табл. 4.2*.

табл 4.2 Получени резултати за интензивностите

λ	150	450	750	1050	1350
λ_1	0	0	50	150	250
λ_2	150	450	700	900	1100

Средното време на чакане на пакети в случай на хетерогенни комуникационни канали и за двата механизма за предаване на пакети се изчислява по известната в теорията на масовото обслужване формула:

(4.16) $$T=\sum p_i t_i,$$

където p_i са вероятностите, получени по формула (4.10), респективно (4.13), а t_i е съответно времето за чакване в M/M/1 система за масово обслужване.

В *табл. 4.3* са показани получените числени резултати за времето за чакане (в μs) при различни интензивности на постъпване на пакетите за механизмите MPT (втората колона) и предложения тук Optimized Packet Transmission Mechanism - OPTM (третата колона).

Средните времена на чакане на пакетите за механизмите MPT и OPTM се увеличават с увеличаване на ρ, което се и очаква според теорията на масовото обслужване.

Трябва да се отбележи, че индексът на намаляване на времето за чакане (стойностите в последната колона на табл. 4.3) е изчислен по:

(4.17) $$Delay\ reduction = (T_{MPT} - T_{OPTM}) / T_{MPT},$$

където T_{MPT} - средно време на изчакване на пакети за MPT, а
T_{OPMT} - средно време на изчакване на пакети за OPTM.

табл 4.3 Резултати за времето за чакане на пакетите

ρ	MPT	OPTM	Delay reduction
0.1	0.00444	0.00286	0.357
0.3	0.00571	0.004	0.3
0.5	0.008	0.0064	0.2
0.7	0.01333	0.01142	0.143
0.9	0.04	0.03555	0.111

От сравнението на резултатите може да се направи извода, че предложеният тук алгоритъм (OPTM) е по-добър от MPT, тъй като позволява да се намали времето за чакане на пакетите при хетерогенни комуникационни канали между 11.1 и 35.7%.

За съжаление, трафикът в Интернет (включително и IoT) днес не е Поасонов. Ето защо формула (4.2) не е приложима, следователно не е приложим и методът на Лагранж с неопределените коефициенти. По-долу се предлага използването на еволюционни методи на изкуствения интелект за реализиране на оптимизацията.

Оптимизация чрез Генетични Алгоритми

Генетичните алгоритми са част от еволюционото програмиране, което е бързо разрастващата се област на изкуствения интелект. Генетичните алгоритми са вдъхновени от дарвиновата теория за еволюцията. Следователно, решенията на проблеми с генетични алгоритми са еволюционни.

Всички живи организми се състоят от клетки. Във всяка клетка има множество хромозоми. Хромозомите са низове от ДНК и служат като модел на целия организъм. Хромозомите се състоят от гени, блокове на ДНК.

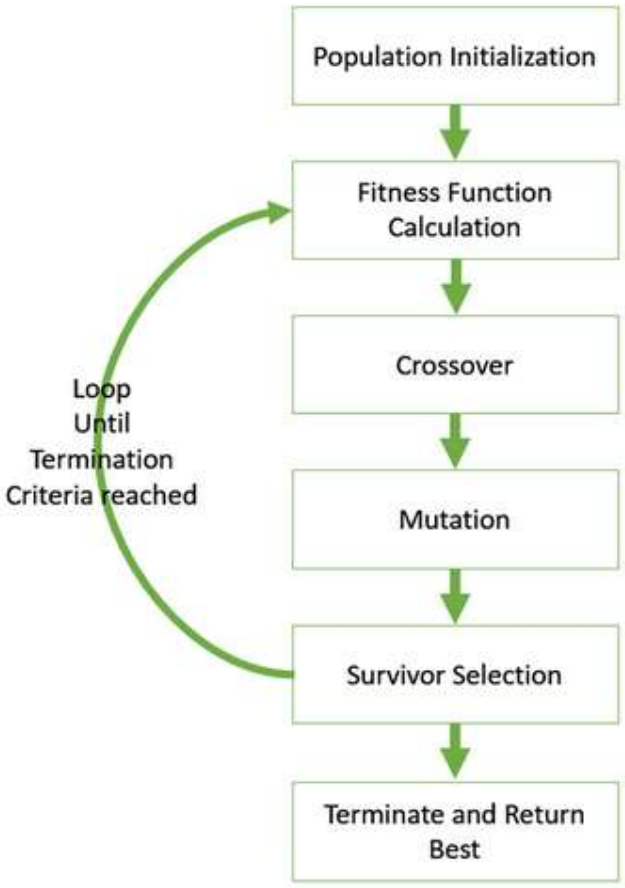
При репродукция, първо се появява рекомбинация. Гените от родителите формират по някакъв начин изцяло нова хромозома. Новосъздаденото потомство след това може да бъде подложено на мутация. Мутация означава, че елементи от ДНК леко се променят. Тези промени главно са породени от грешки при копирането на гените от родителите.

Алгоритъмът се стартира с множество от решения (представени от хромозоми) наречени популация. Решенията от една популация се вземат и

използват от новта популация. Очаква се, че новата популация ще бъде по-добра от старата. Това се повтаря докато някакво условие се удовлетвори (например, достигане на определен брой популации или достигане на най-доброто/съботимално решение).

С това как да се създадат хромозомите и какъв тип кодиране се използва са свързани кръстосването и мутацията (двете основни операции на GA). Следва избирането на родители за кръстосване. Един начин е като се избират по-добри родители (с надеждата, че те ще създадат по-добро поколение).

Последователността от стъпки за Основния Генетичен Алгоритъм са дадени по-долу на **фиг. 4.4**.



Генериране на случайна популация от n на брой хромозоми (подходящи решения на проблема).

Изчисляване жизнеспособността f(x) на всеки хромозом в популацията. Създава се нова популация. Избират се два родителски хромозома

При кръстосването вероятностно се кръстосват родителите за да формират новото поколение (деца).

При мутацията вероятностно мутира новото поколение на някое място в хромозомата.

[Приемане+Заместване] Поставяне на новото поколение в новата популация. Използване на ново-генерираната популация за по-нататъшни изпълнения на алгоритъма

Ако крайното условие е удовлетворено, спиране и връщане на най-доброто решение в настоящата популация

фиг. 4.4 Обобщена блок-схема на GA

Тук е даден разгледания по-горе числен пример с използване на Solver на MS Excel като калкулатор на генетични алгоритми. Численият експеримент се провежда със същите входни данни.

Честотната лента на виртуалните канали (b_1 и b_2) са константи, поради което те не са избрани да са хромозоми. Те са дефинирани в таблицата на MS Excel.

Вероятностите за препращане на пакетите през съответните канали p_i i ϵ {1, 2… N} са избрани да са хромозоми. Последните варират между 0 и 1 и определят интензивностите, които ще бъдат съответно $\lambda_i = p_i*ro*b$, както и времето за

изчакване на пакетите. При конкретни стойности p_i i ϵ {1, 2… N} се получава минимум на времето за изчакване на пакетите при прехвърляне на пакетите във виртуалните канали (оптималният резултат).

Генетичният алгоритъм работи чрез сравняване на най-добрите (Fitness) стойности на съседните поколения на популацията. Целевата функция е представена чрез формула в таблицата на MS Excel и дава времето за изчакване - виж формула (4.16):

$$\frac{p1}{b1-(p1*ro*b)}+\frac{p2}{b2-(p2*ro*b)} \rightarrow \text{MIN} \quad (4.18)$$

Трябва да се отблежи, че целевата функцията връща времето на чакане само, когато са изпълнени условията (constraints в калкулатора на ГА):

1) всяко от събираемите в (4.18) трябва да е неотрицателно и

2) общата вероятност трябва да бъде 1 ($\sum p_i=1$), 0< p_i<1.

Резултатите, получени с калкулатора на генетични алгоритми Solver, за средното време на изчакване за OPTM са показани съответно в *табл 4.4* и *табл 4.5*. Както може да се види от *табл 4.4*, времената за изчакване се увеличават при увеличаване натоварването на системата, което е логично, като се има предвид теорията на масовото обслужване.

табл 4.4 Числени резултати за оптимални вероятности

ro	p_1	p_2	Delay, ms
0,1	0	1	0.002857143
0,3	0	1	0.004
0,5	0.06666	0.93333	0.0064
0,7	0.14285	0.85714	0.011428571
0,9	0.18516	0.81483	0.035555556

В *табл 4.5* получените резултати за времената за изчакване се сравняват с числените резултати, получени по-рано (чрез метода на Лагранж с неопределени коефициенти) за същия оптимизиран механизъм за препращане на трафика по множество хетерогенни комуникационни канали (Optimized Packet Transmission with Multipaths - OPTM) със същите входни данни.

табл 4.5 Числени резултати за времето на изчакване

ro	delay	GA	err
0.1	0.002857143	0.002857143	0
0.3	0.004000000	0.004000000	0
0.5	0.006400000	0.006400000	1.6575E-13
0.7	0.011428571	0.011428571	1.1683E-10
0.9	0.035555556	0.035555564	2.276E-07

Резултатите, получени при използване на калкулатора на генетичните алгоритми и тези по метода на Лагранж са почти идентични. Най-голямата разлика се наблюдава при натоварване на системата 0,9 (от порядъка 10^{-7}), което потвърждава коректността на предложения метод за минимизиране на времето за предаване на пакети в IoT.

Глава 5. Разработване на система за симулиране на кибератаки и събиране на данни за мрежовия трафик, натоварването на процесора и паметта

Предложените до тук подходи изискват събирането на данни за използване на паметта, мрежовия комуникационен интерфейс и процесора на не/компрометирани IoT устройства при кибератаки.

Целта на настоящата глава е създаването на прототип на система за симулиране на кибератаки и събиране на данни за мрежовия трафик, натоварването на процесора и паметта. Основните функционалности на системата са:

- събиране и съхраняване на данни от крайните мрежови устройства (хостове) за мрежовия трафик, натоварването на процесор и памет в централно хранилище за последващо обработване и класифициране на същите като компрометирани или некомпрометирани вследстствие на кибератаки;
- симулиране на кибератаки и визуализиране и обобщаване на цялата събрана информация при наличие или отсъствие на кибератаки;
- използване на централизиран модел за управление, комуникация и съхранение на данни в системата;
- удобен и лесен за употреба уеб интерфейс, който ще служи за основен панел за управление и наблюдение на мрежата и хостовете.

Предложената система се състои от клиентска машина - «мишена», атакуваща машина и машина за събиране на данни. За създаването на трите виртуални машини и виртуален комутатор, чрез който те са свързани в локална мрежа се използва VMware ESXi hypervisor. Избраният подход позволява разширяването на разработваната система с виртуални машини (крайни устройства), като например IoT модули, мрежови комуникационни устройства (различни рутери и stateful и stateless защитни стени, налични като готови за импортиране виртуални машини от Cisco, Huawei и др).

Разработваната система ще позволява симулиране на кибератаки и събиране на данни за натоварването на мрежовия интерфейс, паметта и процесора на IoT устройства както при отсъствие, така и при състояща се кибератака.

Анализ на системи за осигуряване на сигурност в мрежите

За осигуряване на сигурност на мрежите се използват защитни стени (firewalls), Intrusion detection system (IDS) и Intrusion prevention system (IPS) системи. Основната разлика между тях е, че защитната стена изпълнява действия като блокиране и филтриране на трафика, докато IPS и IDS откриват и предупреждават системния администратор за наличие на нестандартни събития, или предотвратяват опити за пробив в зависимост от конфигурацията.

Защитната стена позволява или отхвърля трафик въз основа набор от конфигурирани правила. Разчита се на сокети (адресите на източника и дестинацията, и портовете). Защитната стена блокира всеки трафик, който не отговаря на конфигурираните правила. Системата за откриване на проникване (IDS) е пасивно устройство или приложение, което наблюдава пакетите, изпращани по мрежата, сравнявайки ги с определени шаблони и задейства аларма при откриване на подозрителна дейност. Системата за предотвратяване на проникване (IPS) от друга страна е активно устройство, надграждащо IDS с възможност за предотвратяване на атаките, като ги блокира.

На базата на направен анализ тук се проектира и демонстрира работата и приложението на такава IDS система за осигуряване сигурност в мрежата, която няма недостатъците на анализираните системи.

Разработване на системата за симулиране на кибератаки и събиране на данни

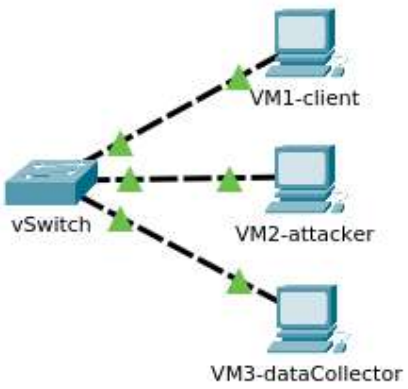
По-долу е представено изграждането на система за събиране на данни за мрежовия трафик, натоварването на процесора и паметта от клиентски машини.

Изградената система се състои от 3 виртуални машини: клиентска машина, атакуваща машина и машина за събиране на данни.

Трите виртуални машини са свързани към един виртуален комутатор. Топологията на предлаганата система е показана на **фиг. 5.7**.

За създаването на трите виртуални машини и виртуалния комутатор, чрез който те са свързани в локална мрежа се използва VMware ESXi hypervisor.

На базата на анализ на различни инструменти за наблюдение на мрежата е избран инструментът Cacti. Cacti е безплатен широко използван инструмент, с удобен графичен интерфейс (GUI) и богати възможности за събиране на данни за мрежовия трафик, натоварването на процесора и паметта на клиентските машини в мрежата. Cacti използва протокола SNMP - Simple Network Management Protocol. За да работи коректно Cacti, устройствата в мрежата следва да са с активиран SNMP агент и да има отделен сървър за централизирано наблюдение, където Cacti събира системните данни от тези устройства.



фиг. 5.7 Топология на предлаганата система

И трите виртуални машини са с операционна система lubuntu 16.04 и имат следните параметри:

-1 процесор;

-1ГБ RAM памет;

-10ГБ диск;

-виртуален мрежови интерфейс 1Gbps.

Първоначално е създадена клиентската виртуална машина, като е инсталирана операционната й система. След това същата е актуализирана чрез командата:

$ sudo apt-get update && sudo apt-get upgrade

Накрая, използвайки инструмента VMware vCenter Converter Standalone клиентската виртуална машина е копирана 2 пъти - първо за атакуваща машина и след това за машина за събиране на данни.

Клиентска машина „мишена“

В системата се използва инструментът Cacti за събиране на данни за мрежовия трафик, натоварването на процесора и паметта на клиентските машини. Cacti използва протоколът SNMP, за да обменя с клиентската машина („мишена“) тези данни. Ето защо на „мишената“ е необходимо да е активиран SNMP.

Първо, на клиентската машина се инсталират пакетите snmp и snmpd чрез командата:

$ sudo apt-get update && sudo apt-get install snmp snmpd

След това се променя конфигурационният файл на SNMP агента като се използва текстовият редактор nano. В [ПР4] е даден конфигурационният файл, използван в настоящата работа, където са добавени редове, свързани с поведението на агента, работещ чрез SNMP протокола.

Атакуваща машина

Както бе споменато по-горе, атакуваща машина е копие на клиентската виртуална машина, като за копирането е използван инструментът VMware vCenter Converter Standalone. Тук за симулиране на различни видове атаки, се използва терминалният инструмент hping3, тъй като е широко разпространен и лесен за употреба. Ето защо, допълнително върху атакуваща машина е инсталиран hping3 пакетът, което става чрез командата:

$ sudo apt-get update && apt-get install hping3

По-долу се симулират различни видове атаки като чрез терминалния инструмент hping3 се генерира съответния зловреден мрежов трафик. Така симулираните атаки са извършени само и единствено в рамките на настоящата работа.

Машина за събиране на данни

Както бе споменато по-горе Cacti е един от най-широко използваните инструменти за управление на мрежата. Той е с отворен код, има вградени функции за аутентикация на потребители и има вградени шаблони за графики като най-често използваните са за честотна лента, използване на дискове, натоварване на процесор/процесорните ядра, статистика за използването на RAM паметта и други. Инсталирането на Cacti и необходимите за работата му пакети (MySQL, PHP, RRDTool, net-snmp и уебсървър, който поддържа PHP, например Apache) става чрез командата:

$ sudo apt-get update && apt-get install cacti

След инсталирането, уеб интерфейсът на Cacti се достъпва чрез URL: http://192.168.0.160/cacti , където 192.168.0.160 е IP адресът на машината за събиране на данни.

Конфигуране на Cacti

Cacti събира информация от устройства чрез SNMP агента, конфигуриран предварително с конфигурационния файл snmpd.conf.

Клиентската машина се добавя към устройствата, за които Cacti събира данни и се задават параметрите й.

За визуализиране на събраните данни първо трябва да се създаде шаблон за графиките, които ще бъдат генерирани от извлечените данни. След това с натискането на бутона “Create graphs for this host” се създава графика и за целта на настоящата работа се избира мрежовата карта, за която ще се следи трафикът. Аналогично се създават графики за натоварването на процесора и RAM паметта.

Следва създаване на дърво от графики и задаване на параметрите на това дърво. Накрая, към дървото с графики се добавят графики за мрежовата карта, на която ще се следи трафика, натоварването на процесора и RAM паметта.

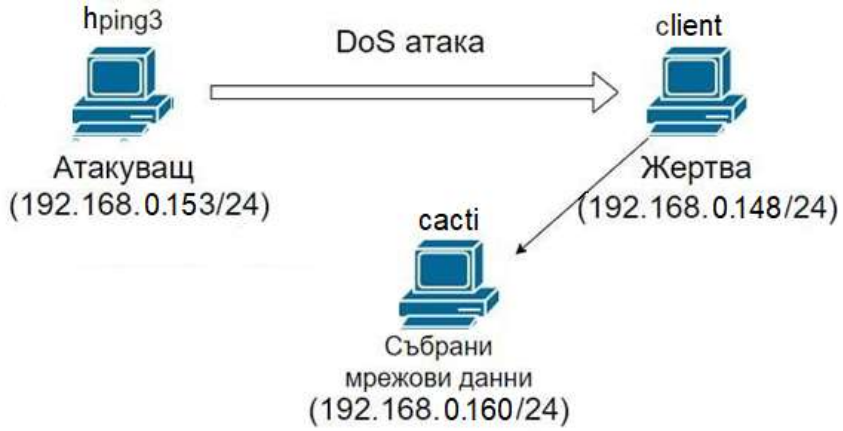
Графиките от събраните по SNMP данни могат да бъдат визуализирани чрез натискане на бутона “Graphs” в горния ляв ъгъл на уеб интерфейса. Също така от тази страница данните могат да бъдат свалени в CSV формат за по-нататъшна обработка.

Симулиране на атаки и получени резултати от работата на системата

За целите на настоящата работа е генериран зловреден мрежов трафик като са извършени симулации на три вида DoS атаки в контролирана среда. Трите типа DoS атаки са SYN Flood, ICMP Flood и UDP Flood. Симулираните тук атаки са извършени само и единствено с цел извличане на необходимата системна информация от тях. По-долу е дадено описание на стъпките за извършване на трите вида DoS атаки.

За нуждите на симулациите на DoS атаки са използвани двете машини: едната, от които играе ролята на атакуващ, а другата – ролята на мишена. Атакуващата машина използва инструмента hping3 за симулиране на атаки.

На **фиг. 5.10** е дадена топологията на системата за извършване на симулация на атака. Атакуващата машина има IP адрес 192.168.0.153 и 24-битова маска, а машината жертва – 192.168.0.148 и 24-битова маска. Събирането на мрежови данни (**фиг. 5.10**) се осъществява от третата машина, на която е инсталиран и конфигуриран инструментът Cacti. Нейният IP адрес е 192.168.0.160 и 24-битова маска.



фиг. 5.10 Топология на системата за симулиране на атаки и събиране на мрежови данни

За стартиране на отделните атаки се използва терминалният прозорец, в който се въвеждат съответните команди.

За симулиране на SYN flood атаката е използвана командата:

hping3 -c 5000 -d 120 -p 80 —flood –rand-source 192.168.0.148

За симулиране на ICMP flood атаката е използваната команда е:

hping3 -c 5000 -d 120 –icmp –flood –rand-source 192.168.0.148

За симулиране на UDP flood атаката е въведена командата:

hping3 -c 5000 -d 120 –udp –flood –rand-source 192.168.0.148,

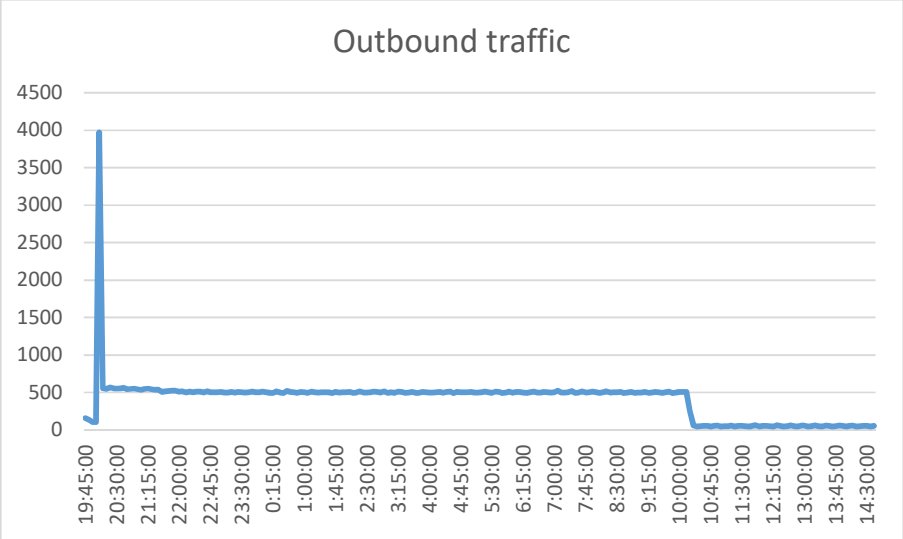
където 192.168.0.148 е IP адресът на получателя на пакетите („мишената“). Използваните параметри при изпълнението на всяка една от командите са както следва:

-c - определя броя генерирани пакети;
-d - определя големината на генерираните пакети;
-p - задава порта на мишената, към който да бъдат насочени пакетите;
--flood - флаг, който указва пакетите да бъдат изпращани възможно най-бързо;
--rand-source - задава случаен IP адрес на изпращача на всеки пакет.

Командите се изпълняват множество пъти с различни параметри, като се променят „брой генерирани пакети“ и „големина на генерирани пакети“, за да се създаде разнообразно множество от събрани данни.

На **фиг. 5.11** са показани първите получени данни за натоварването на мрежовия интерфейс на машината «мишена» в резултат на ICMP flood атака, стартирана в 20:05 часа и приключваща в 10:10 часа. Видна е разликата в натоварването на мрежовия интерфейс на машината «мишена», при нормален (преди 20:05 и след 10:10) и зловреден мрежови трафик (по време на реализиране на атаката - между 20:05 и 10:10).

Поради ограничения обем на автореферата получените резултати от работата на системата (системни данни), т.е. данните за натоварването на мрежовия интерфейс, процесора и паметта на машината «мишена» в резултат на SYN Flood, ICMP Flood и UDP Flood атаки са дадени в [ПР4].

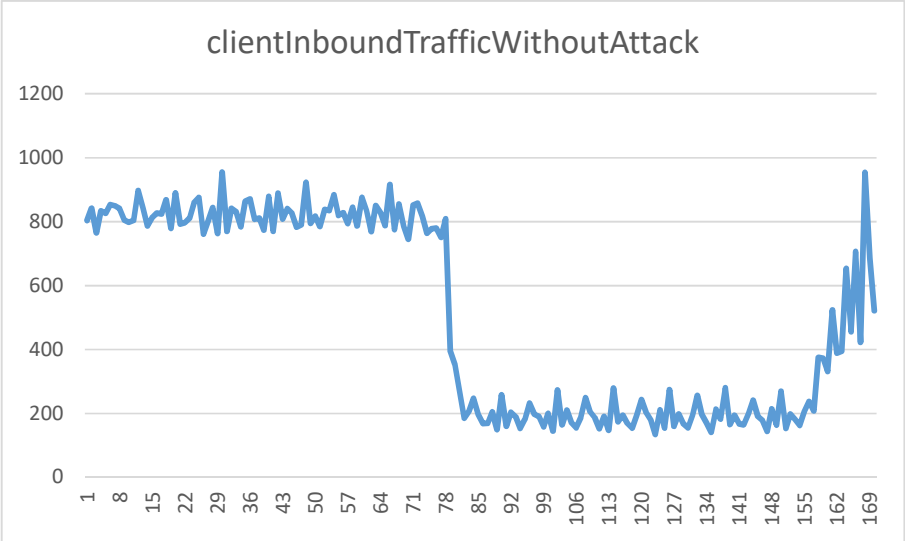


фиг. 5.11 Нормален мрежов трафик и такъв при реализиране на атака

Събраните системни данни могат да бъдат използвани за последваща обработка чрез достатъчно точен и бърз алгоритъм, като този в глава 2, който да разпознава „нормалния“ трафик и да класифицира разнообразните видове DoS и DDoS атаки.

Трябва да се отбележи, че за IPS/IDS системите от най-голямо значение са данните за мрежовия трафик. Получените системни данни за входящия (Inbound) и

изходящия (Outbound) мрежов трафик към/от машината жертва, в случаите, когато трафикът към/от нея е „нормален“ и при атакуването ѝ чрез ICMP Flood са дадени като .CSV файлове със съответните имена в [ПР4]. На **фиг. 5.12** са показани графично системни данни за входящия мрежовия трафик.



фиг. 5.12 Графично представени времеви редове за мрежов трафик

Разработен е авторски скрипт на Python [ПР4] реализиращ Wavelet трансформация за събраните системни данни от машината жертва. Резултатите от работата на Python скрипта върху данните за общия мрежов трафик (Inbound+Outbound) към/от машината жертва, в случаите, когато трафикът към нея е „нормален“ и при ICMP Flood атака са показани в *табл. 5.2*. В първия ред е дадена енергията - E на „нормалния“ трафик, а във втория ред - енергията - E при провеждането на ICMP Flood атака. Таблицата е допълнена с трети ред, в който е дадена разликата в проценти между енергията E за нормален трафик и при провеждането на ICMP Flood атака.

табл. 5.2 Резултати за общия мрежов трафик

Вид мрежов трафик	**E**
Нормален трафик	7233
Трафик при ICMP Flood атака	12691
Разлика в енергията E	43 %

Както е видно от *табл. 5.2* разликата в енергиите за нормален трафик и при провеждането на ICMP Flood атака е 43%, което потвърждава целесъобразността на използването на Wavelet алгоритъма, респективно предложения Python скрипт.

Детектор на кибератаки с модул за логистична регресия

Логистичната регресия е част от машинното обучение, използвана за решаване на задачи за двоична класификация, където целевата променлива е

дихтомична. Следвайки последните тенденции в откриването на мрежови атаки, насочени към IoT, някои от които описани по-горе (Глава 1), и по-нататъшно развитие на предходно направените изследвания (Глави 2 и 3), в които са предложени и тествани единични детектори за разпознаване на компрометирани IoT устройства, и симулирането на различни кибератаки, по-долу в настоящата глава се предлага изследването на конкретен комбиниран детектор.

При логистична регресия се изчислява прогнозната вероятност (от 0 до 1) да бъде реализирана конкретната категория.

Числен експеримент за изследване на точността на детектиране

Целта е да се изследва точността на детектиране на компрометирани устройства при кибератаки чрез свързването на различни детектори към модул за логистична регресия.

Целевата променлива показва наличието/отсъствието на атака, като числото 0 означава, че този запис е при нормален трафик, а 1 – при наличието на атака. Признаците и тяхното разпределение са описани по-долу.

Всеки отчет от набора с данни съдържа следните 10 признаци като числови стойности: stime – начално време на регистриране на записа, pkts – брой пакети, които включва записа, dur - продължителност, mean - среден период на връзка за генериране на определен запис, stddev - стандартно отклонение на отчетите след агрегиране, sum - време за съществуване на връзките за записа, min - минимална стойност за времето на съществуване на регистрираните записи, max - времето на съществуване на връзките, за които са получени най-продължителните записи и rate - скорост на пакетите при обмен между източника и получателя.

На **фиг. 5.16** е показан работен екран от Jupyter Notebook, на който се виждат предсказаните стойности за целевата функция и индексите на записите, за които е предсказано случването на атака, и че точноста е 0.888889.

```
In [71]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=0)

In [72]: classifier = LogisticRegression(solver='lbfgs',random_state=0)

In [73]: classifier.fit(X_train, Y_train)
Out[73]: LogisticRegression(random_state=0)

In [74]: predicted_y = classifier.predict(X_test)
         predicted_y
Out[74]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])

In [75]: for x in range(len(predicted_y)):
             if (predicted_y[x] == 1):
                 print(x, end="\t")
         0       1       2       3       4       5       6       7       8       9       10      11      12      13      14
                 15      16      17      18      19      20      21      22      23      24      25      26      27      28
                 29      30      31      32      33      34      35

In [76]: print('Accuracy: {:.6f}'.format(classifier.score(X_test, Y_test)))
         Accuracy: 0.888889
```

фиг. 5.16 работен екран от Jupyter Notebook

От поучените резултати следва да се заключи, че се постига висока точност на откриване на компрометирани след тестове с популярния набор от данни Bot-IoT. Предложеният детектор е по-точен от други решения, предложени от други автори. В литературата се анализират и сравняват алгоритъма случайна гора от дървета за вземане на решения (Random Forest- RF) с плътните невронни мрежи (Dense Neural Network), които постигат почти еднаква точност (precision) - от 0,87, но RF изисква

по-малко отчети. Експеримент с набор от данни за по-разнообразни видове атаки показва малко по-ниска точност на RF алгоритъма, но все пак той постига по-висока точност - от 0,77 пред Naive Bayes (NB), а алгоритъмът ID3 постига по-висока такава - съответно 0,78.

Втори експеримент за изследване на точността на детектиране

Численият експеримент тук е реализиран като към записите за трафика без атака и при наличие на такава допълнително се включват и стойностите за настроението (mood) от детектора базиран на емоционални модели, както и други данни за мрежовия трафик от наличния DataSet (bytes - брой байтове, които включва записа, ltime - крайно време на регистриране на записа, seq - идентификатор на последователността на запис, spkts - брой пакети от източника, които включва записа, dpkts - брой пакети от получателя, които включва записа, sbytes - брой байтове от източника, които включва записа, dbytes - брой байтове от получателя, които включва записа, srate - скорост на пакети от източник към получател и drate - скорост на пакети от получател към източник).

Използват се времевите редове на натоварването на процесор, памет и мрежов интерфейс при злонамерен и нормален трафик за EMO детектора на компрометирани IoT устройства.

На **фиг. 5.17** е показан работен екран от Jupyter Notebook, на който се виждат предсказаните стойности за целевата функция и индексите на записите, за които е предсказано случването на атака. Също така на фигурата се вижда, че точноста е 0.944444.

```
In [82]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=0)

In [83]: classifier = LogisticRegression(solver='lbfgs',random_state=0)

In [84]: classifier.fit(X_train, Y_train)
Out[84]: LogisticRegression(random_state=0)

In [85]: predicted_y = classifier.predict(X_test)
         predicted_y
Out[85]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1])

In [86]: for x in range(len(predicted_y)):
             if (predicted_y[x] == 1):
                 print(x, end="\t")
         0       1       2       3       4       5       6       7       8       9       10      12      13      16      17
                 18      19      20      21      22      23      24      25      26      27      28      30      31      32
                 33      34      35

In [87]: print('Accuracy: {:.6f}'.format(classifier.score(X_test, Y_test)))
         Accuracy: 0.944444
```

фиг. 5.17 Работен екран от Jupyter Notebook

От поучените резултати следва да се заключи, че точността е по-висока при увеличаване броя на използваните предиктори (**фиг. 5.16** и **фиг. 5.17**).

III. Приноси в ДТ

По мнението на автора приносите имат основно научно-приложен и приложен характер:

Научно-приложни приноси

1. Анализирани са методи и средства на системите с изкуствен интелект и конкретни инженерни решения в Интернет на нещата- IoT за висока мрежова сигурност и оптимална свързаност.
2. Предложени са метод и два варианта на алгоритъм на метода за интелигентна система за детектиране на компрометирани IoT устройства в резултат на кибератака използваща Wavelet трансформации и филтър на Haar.
3. Предложена е система базирана на емоционален модел за откриване на компрометирани IoT устройства при кибератака, като е разработен калкулатор на емоционални модели и е проведен пълен факторен експеримент, и са оптимизирани параметрите на модела.
4. Предложени са механизъм за препращане на трафика по множество канали - OPTM Optimized Packet Transmission with Multipaths и генетичен алгоритъм за оптимизиране на времето за чакане на пакетите при Multipath свързаност, верифициран чрез метода на Лагранж, като средното време за чакане на пакетите при предложеният механизъм е по-малко с между 11% и 36% спрямо решение на Korean Telecom.

Приложни приноси

5. Предложени са система симулираща различни видове DoS атаки и събираща системни данни при наличие на кибератаки, и детектор използващ модул за логистична регресия, който разграничава нормалния и зловредния трафик на базата на събраните от хостовете системни данни при реализирани атаки, като предложеният детектор е по-точен от подобни решения, предложени от други автори.

IV. Публикации свързани с ДТ

П1. Hristov A., Trifonov R. A Model for Identification of Compromised Devices as a Result of Cyberattack on IoT Devices, Proceedings of the 2021 International Conference on Information Technologies (InfoTech-2021), IEEE Conference, Rec # 52438, 16-17 September 2021, St. St. Constantine and Elena, Bulgaria, http://infotech-bg.com/sites/default/files/proceedings/InfoTech-2021-Proceedings.rar

П2. Христов А. Интелигентна система за детектиране на компрометирани IOT устройства в резултат на кибератака, Българско списание за инженерно проектиране, ISSN 1313-7530, брой 43, януари 2021г., стр. 17-22, https://bjed.tu-sofia.bg/items/BJED-0043(2021).pdf

П3. Hristov A., Trifonov R. Using web based calculator of emotional models for identification of compromised internet of things devices, Българско списание за инженерно проектиране, ISSN 1313-7530, брой 44, ноември 2021г., стр. 13-18, https://bjed.tu-sofia.bg/items/BJED-0044(2021).pdf

П4. Hristov V., Hristov A. Optimizing Packet Transmission Mechanism with Multipath Technologies, Proceedings in the International Scientific Conference “Computer Science'2022”, IEEE Conference, Rec #55378, 30 May - 2 June 2022, Sofia, Bulgaria

П5. Hristov A., Trifonov R. System for simulating cyberattacks and data acquisition for network traffic and usage of processor and memory of hosts, Proceedings of International Scientific Conference "Engineering. Technologies. Education. Safety, 2022", Publisher: Scientific technical union of mechanical engineering “Industry - 4.0”, ISSN 2535-0315, pp. 39-42, https://techtos.net/sbornik/1-2022.pdf

П6. Христов А., Трифонов Р. Симулиране на някои конкретни видове кибератаки, Българско списание за инженерно проектиране, ISSN 1313-7530, брой 45, 2022г., с. 5-10, https://bjed.tu-sofia.bg/items/BJED-0045(2022).pdf.

Забелязано цитиране в Scopus на публикация [П1] свързана с ДТ

Romansky, R. Formalization and Investigation of Parallel Processes Dispatching, Proceedings of the 2022 International Conference on Information Technologies (InfoTech-2022), IEEE Conference, Rec # 55606,September 2022, DOI: 10.1109/InfoTech55606.2022.9897104

V. Приложения

Приложенията с разработените авторски Python скриптове, програма на C, алгоритми и данни са качени в хранилища (repositories) в github на автора.

ПР1 https://github.com/sashkinaaa/readValuesFromBMP
ПР2 https://github.com/sashkinaaa/Haar_1D_Filter
ПР3 https://github.com/sashkinaaa/EmotionalModel
ПР4 https://github.com/sashkinaaa/cacti
ПР5 https://github.com/sashkinaaa/LogisticRegression
ПР6 https://www.researchgate.net/profile/Alexander-Hristov-4/publication/367220583_getMoodOutput/data/63c7b805e922c50e99a30d23/getMoodOutput.7z

VI. ABSTRACT

This PhD thesis provides a solution to such a problem as identifying compromised Internet of Things (IoT) devices, starting with an analysis of the literature (state of the art) and defining the goal and tasks to achieve it.

The purpose of the second chapter is to propose an intelligent system for identification of compromised IoT devices due to cyberattack, using Wavelet transformation and Haar filter. Monitoring is being made and the state is identified through time-synchronized series of indexes for usage of processor, memory and network interface card. Using system monitor the time-synchronized indexes of non-compromised IoT devices as well as indexes of compromised due to some well-known cyberattacks IoT devices are saved. The parameters of the proposed system are being specified in order to distinguish (filter) the two states of the IoT devices (non-compromised or compromised) by these indexes.

Third chapter aims to propose another approach for identification of compromised IoT devices due to cyberattack, using Emotional Models (EM). The parameters of the proposed system are being specified in order to detect the compromised and non-compromised states of the IoT devices by the indexes for usage of processor, memory and network interface card. A web based calculator of EM is presented, and its usefulness is discussed. Finally, the parameters of the EM model are optimised.

Forth chapter is dedicated to propose a mechanism for traffic allocation over numerous heterogeneous communication channels (Optimized Packet Transmission with Multipaths – OPTM). It allows minimizing waiting time of packets for different Multipath technologies. For this purpose, an analysis of different Multipath technologies has been made. A model for minimizing the waiting time of packets using MS Excel Solver as a calculator of genetic algorithms is developed for a packet forwarding mechanism. Numerical experiment has been conducted for the proposed mechanism with input data for MPT from a Korean Telecom commercial solution. By comparing the obtained results for MPT and OPTM mechanisms, it can be concluded, that the proposed here packet transmission mechanism is better, because the mean waiting times for OPTM are better than these for MPT by 11-36%.

Final chapter aims to create a prototype system for simulating cyberattacks with the ability to collect and store data from the IoT devices(hosts) for network traffic, processor and memory usage in a central repository for subsequent processing and classification of the hosts as compromised or non-compromised. The system uses a centralized model for management, communication and storage of data and has a user-friendly web interface for managing and monitoring the hosts, and visualizing and summarizing all the information collected when cyberattacks are present or absent.