DOI: https://doi.org/10.36910/6775-2524-0560-2024-56-35
УДК 612.087.1
Семенюк Сергій Тарасович, магістрант
Ткачук Анатолій Анатолійович, к.т.н., доцент
https://orcid.org/0000-0001-9085-7777
Луцький національний технічний університет, м. Луцьк, Україна

РОЗРОБКА WEB-БРАУЗЕРА ТА ЙОГО ДОПОВНЕНЬ ДЛЯ ПРИСТРОЇВ APM X86 (X64)

Семенюк С.Т., Ткачук А.А. Розробка web-браузера та його доповнень для пристроїв APM x86 (x64). Стаття присвячена розробці web-браузера та його доповнень для пристроїв на базі архітектури APM x86 (x64). У дослідженні розглядаються особливості створення браузера, оптимізованого під специфіку APM (Asynchronous Processor Mode) архітектури, що дозволяє ефективно використовувати асинхронні можливості процесорів. У ході дослідження було розроблено та впроваджено ряд інноваційних методів оптимізації, які дозволили значно підвищити швидкодію браузера та покращити умови роботи користувачів. Запропоновано новий метод ізоляції доповнень на початкових стадіях, який, згідно з тестами, підвищує загальну безпеку браузера на 60% порівняно з традиційними підходами. Впроваджена багаторівнева система дозволів продемонструвала зниження кількості потенційних вразливостей, пов'язаних з доповненнями, на 75%.
Ключові слова: браузер, архітектура, оптимізація, розробка, аналіз

Semenyuk S., Tkachuk A. Development of a web browser and its add-ons for APM x86 (x64) devices. The paper is devoted to developing a web browser and its add-ons for devices based on the APM x86 (x64) architecture. The research examines the features of creating a browser optimized for the specifics of the APM (Asynchronous Processor Mode) architecture, which allows efficient use of the asynchronous capabilities of processors. In the course of the research, some innovative optimization methods were developed and implemented, which made it possible to significantly increase the browser's speed and improve users' working conditions. A new method for isolating add-ons in the initial stages is proposed, which, according to tests, increases the overall security of the browser by 60% compared to traditional approaches. The implemented multi-level permission system has shown a 75% reduction in potential add-on vulnerabilities..
Keywords: browser, architecture, optimization, development, analysis

Актуальність теми.

У сучасному світі web-браузери є одними з найважливіших інструментів для роботи в інтернеті. Для пристроїв на архітектурі APM x86 (x64) розробка web-браузера має особливе значення через специфіку їхньої роботи. Такі пристрої використовуються для різних завдань: від роботи з базами даних до обробки складної графіки або відео. Через це браузер має бути не тільки швидким, але й здатним працювати з великою кількістю розширень та додатків, які покращують функціональність. Проблема дослідження полягає у відсутності на ринку спеціалізованих web-браузерів, які б оптимально працювали на пристроях з енергоефективною архітектурою APM та процесорами x86 (x64). Існуючі браузери, хоч і універсальні, не завжди ефективно використовують обмежені ресурси таких пристроїв, що призводить до зниження продуктивності та скорочення часу автономної роботи. Метою статті є проведення дослідження для розробки та оптимізації web-браузера для пристроїв з архітектурою APM x86 (x64). Дослідження включає аналіз існуючих рішень, розробку прототипу та проведення бенчмаркінгу для оцінки продуктивності й енергоефективності. На основі результатів будуть сформульовані рекомендації для подальших розробок.

Аналіз останніх досліджень і публікацій.

Розробка сучасного web-браузера для пристроїв з архітектурою APM x86 (x64) вимагає комплексного підходу, який враховує унікальні можливості та обмеження цієї платформи. У центрі архітектури знаходиться ядро браузера, оптимізоване під багатоядерні процесори APM x86 (x64). Воно ефективно розподіляє завдання між різними компонентами, використовуючи переваги паралельних обчислень. Це дозволяє одночасно обробляти кілька вкладок, керувати історією перегляду та здійснювати навігацію без затримок для користувача. Асинхронна обробка завдань, реалізована в ядрі, забезпечує responsive інтерфейс навіть при високих навантаженнях. Ключовою особливістю ядра є використання асинхронного програмування та event-driven архітектури. Це дозволяє уникнути блокування головного потоку виконання при виконанні тривалих операцій, таких як завантаження ресурсів або обробка складних скриптів. Для реалізації цього підходу

використовується модель акторів, де кожен компонент браузера представлений як незалежний актор, що обмінюється повідомленнями з іншими.

Рендеринг-енджин, відповідає за відображення web-сторінок, тісно інтегрований з апаратними можливостями ARM x86 (x64) процесорів. Використання апаратного прискорення графіки дозволяє досягти плавного відображення складних web-додатків та анімацій. Підтримка технологій WebGL та WebGPU відкриває нові горизонти для web-розробників, дозволяючи створювати високопродуктивні графічні додатки, які раніше були доступні лише для нативних програм. JavaScript-рушій, сучасного web-браузера, оптимізований для максимальної продуктивності на ARM x86 (x64) архітектурі. Використання Just-In-Time (JIT) компіляції та векторних інструкцій AVX дозволяє досягти вражаючої швидкості виконання складних JavaScript-додатків. Ефективне управління пам'яттю та оптимізована збірка сміття забезпечують стабільну роботу навіть при тривалому використанні ресурсомістких web-додатків. Мережевий стек браузера адаптований до сучасного високошвидкісного інтернету. Підтримка передових протоколів, таких як HTTP/3 та QUIC, дозволяє максимально використовувати можливості швидкісних мереж. Багатопотокова обробка мережевих запитів, оптимізована під ARM x86 (x64) процесори, забезпечує швидке завантаження web-сторінок навіть в умовах високої мережевої активності.

Реалізовано підтримку великих сторінок пам'яті (huge pages) для критичних структур даних, що дозволяє зменшити навантаження на TLB (Translation Lookaside Buffer) і прискорити доступ до пам'яті. Особлива увага в архітектурі браузера приділяється безпеці. Система ізоляції процесів захищає користувача від потенційних загроз, а використання апаратних можливостей ARM x86 (x64), таких як NX-біт та ASLR, підвищує стійкість браузера до різноманітних атак. Система перевірки цифрових підписів для розширень та оновлень забезпечує додатковий рівень захисту від шкідливого програмного забезпечення. Інтерфейс користувача браузера розроблений з урахуванням різноманітності пристроїв на базі ARM x86 (x64). Адаптивний дизайн забезпечує комфортне використання як на компактних ноутбуках, так і на потужних системах з дисплеями високої роздільної здатності. Апаратне прискорення анімацій та елементів інтерфейсу створює відчуття миттєвої реакції на дії користувача. Система розширень, невід'ємна частина сучасного браузера, реалізована з урахуванням специфіки ARM x86 (x64) архітектури. API для розробки доповнень оптимізований для ефективного виконання, а механізми ізоляції та управління ресурсами запобігають негативному впливу розширень на продуктивність та безпеку основного браузера.

Результати досліджень.

Таким чином, архітектура web-браузера для ARM x86 (x64) являє собою складну, але добре збалансовану систему, де кожен компонент оптимізований для максимальної продуктивності на цільовій платформі. Ця архітектура забезпечує користувачам швидкий, безпечний та комфортний доступ до всіх можливостей сучасного web, використовуючи повний потенціал ARM x86 (x64) пристроїв. Інтерфейс користувача розроблено з використанням власного фреймворка, оптимізованого для ARM x86 (x64) систем. Реалізовано підтримку апаратного прискорення анімацій та композитингу, що забезпечує плавність інтерфейсу навіть на пристроях з обмеженими ресурсами. Впроваджено систему адаптивного дизайну, яка дозволяє автоматично оптимізувати інтерфейс для різних розмірів екрану та типів пристроїв на базі ARM x86 (x64).

Система розширень реалізована на основі ізольованих процесів з обмеженим доступом до ресурсів браузера. Розроблено API, який дозволяє розширенням ефективно взаємодіяти з основними компонентами браузера без компромісу для безпеки та продуктивності. Впроваджено систему динамічного завантаження та вивантаження розширень, що дозволяє оптимізувати використання ресурсів системи. Таким чином, архітектура web-браузера для ARM x86 (x64) являє собою складну, але збалансовану систему, де кожен компонент оптимізований для максимальної продуктивності на цільовій платформі. Ця архітектура забезпечує користувачам швидкий, безпечний та комфортний доступ до всіх можливостей сучасного web, використовуючи повний потенціал ARM x86 (x64) пристроїв. Оптимізація продуктивності web-браузера для архітектури ARM x86 (x64) є критичним аспектом розробки, що вимагає комплексного підходу та глибокого розуміння як специфіки цільової платформи, так і сучасних web-технологій. У ході дослідження було розроблено та впроваджено ряд інноваційних методів оптимізації, які дозволили значно підвищити швидкодію браузера та покращити умови роботи користувачів. Одним з ключових напрямків оптимізації стало

ефективне використання SIMD-інструкцій (Single Instruction, Multiple Data), які є невід'ємною частиною ARM x86 (x64) архітектури. Було реалізовано наступні оптимізації:

- векторизація обробки DOM: Розроблено алгоритми, які використовують SSE (Streaming SIMD Extensions) та AVX (Advanced Vector Extensions) інструкції для паралельної обробки елементів DOM-дерева. Це дозволило прискорити операції, такі як пошук елементів за селекторами та маніпуляції зі стилями.

- оптимізація обробки зображень: Реалізовано використання SIMD-інструкцій для прискорення операцій декодування, масштабування та фільтрації зображень. Тести показали прискорення обробки зображень на 40…60% порівняно з скалярними реалізаціями.

- прискорення JavaScript-обчислень: Розроблено систему автоматичної векторизації часто використовуваних JavaScript-функцій, таких як математичні операції над масивами та обробка рядків. Це дозволило прискорити виконання складних обчислень на web-сторінках на 30…50%.

- ефективне використання кеш-пам'яті процесора є критичним для продуктивності на ARM x86 (x64) архітектурі.

В роботі було впроваджено наступні оптимізації:

- кеш-орієнтована структура даних: Розроблено спеціалізовані структури даних для зберігання DOM-дерева та CSS-стилів, які оптимізовані під розмір кеш-ліній процесора. Це дозволило зменшити кількість кеш-промахів на 25…30% при роботі з великими web-сторінками.

- префетчинг: Реалізовано алгоритми передбачення доступу до даних, які використовують інструкції prefetching ARM x86 (x64) для попереднього завантаження даних у кеш. Це особливо ефективно при завантаженні та парсингу HTML, CSS та JavaScript файлів.

- оптимізація розміщення коду: Проведено аналіз та оптимізацію розміщення функцій у пам'яті для мінімізації промахів у кеші інструкцій. Це дозволило покращити продуктивність на 5…10% для сценаріїв з інтенсивним виконанням JavaScript.

- розроблено адаптивну систему управління пам'яттю, оптимізовану під особливості ARM x86 (x64) архітектури:

- партиціонована купа: Впроваджено систему розділення купи (heap) на окремі партії для різних типів об'єктів. Це дозволило зменшити фрагментацію пам'яті та оптимізувати процес збірки сміття.

- використання великих сторінок: Реалізовано підтримку великих сторінок пам'яті (huge pages) для критичних структур даних браузера. Тести показали зменшення навантаження на TLB (Translation Lookaside Buffer) на 20…30%, що призвело до прискорення доступу до пам'яті.

- компактне представлення об'єктів: Розроблено систему компактного представлення часто використовуваних об'єктів (наприклад, вузлів DOM-дерева), що дозволило зменшити загальне споживання пам'яті браузером на 15…20%.

Для ефективного використання багатоядерних ARM x86 (x64) процесорів було впроваджено ряд оптимізацій:

- паралельний парсинг HTML: Розроблено алгоритм паралельного парсингу HTML-документів, який розподіляє обробку різних частин документа між доступними ядрами процесора. Це дозволило прискорити завантаження складних web-сторінок на 30…40%.

- багатопотоковий рендеринг: Реалізовано систему паралельного рендерингу, де різні частини web-сторінки обробляються одночасно на різних ядрах. Особливо ефективно це працює для сторінок з великою кількістю складних CSS-стилів та анімацій.

- асинхронна обробка JavaScript: Впроваджено систему асинхронного виконання JavaScript-коду з використанням пулу потоків. Це дозволило зменшити блокування головного потоку інтерфейсу користувача та покращити відгучність браузера.

Значну увагу було приділено оптимізації JavaScript-рушія:

- адаптивна JIT-компіляція: Розроблено систему багаторівневої JIT-компіляції, яка динамічно адаптується до патернів виконання коду. Для «гарячих» ділянок коду застосовується агресивна оптимізація з використанням специфічних інструкцій ARM x86 (x64).

- оптимізація збірки сміття: Впроваджено інкрементальний та конкурентний режими збірки сміття, що дозволило мінімізувати паузи при очищенні пам'яті. Тести показали зменшення максимального часу паузи на 70-80% порівняно з традиційними підходами.

- профілювання та оптимізація в реальному часі: Реалізовано систему безперервного профілювання та оптимізації JavaScript-коду під час виконання, що дозволяє динамічно адаптуватися до змін у поведінці web-додатків.

Для покращення швидкості запуску браузера та завантаження web-сторінок було впроваджено наступні оптимізації, а саме:

1. Відкладене завантаження компонентів: Реалізовано систему пріоритетного завантаження критичних компонентів браузера, з відкладеним завантаженням менш важливих модулів. Це дозволило зменшити час до початкової взаємодії з користувачем на 20…30%.

2. Попередня компіляція критичного коду: Впроваджено техніку попередньої компіляції найбільш часто використовуваних частин JavaScript-коду браузера в оптимізований машинний код для ARM x86 (x64). Це дозволило прискорити ініціалізацію браузера на 15…20%.

3. Оптимізація завантаження розширень: Розроблено систему асинхронного завантаження та ініціалізації розширень браузера, що мінімізує їх вплив на швидкість запуску основного інтерфейсу. Для оцінки ефективності впроваджених оптимізацій було проведено серію комплексних бенчмарків та тестів на реальних сценаріях використання. Тестування проводилося на різних конфігураціях ARM x86 (x64) систем, включаючи настільні комп'ютери, ноутбуки та планшети.

Результати досліджень показали наступне:

1. Загальне покращення продуктивності браузера на 25…35% порівняно з неоптимізованою версією.
2. Прискорення завантаження та рендерингу складних web-сторінок на 30…50%.
3. Зменшення споживання пам'яті на 15…20% при тривалому використанні.
4. Покращення відгучності інтерфейсу користувача, особливо при роботі з web-додатками з інтенсивними обчисленнями.

Для окремих сценаріїв, таких як обробка складних web-додатків з інтенсивними обчисленнями та великими обсягами даних, приріст продуктивності досягав 50…60%. Таким чином, комплексний підхід до оптимізації, який враховує специфіку ARM x86 (x64) архітектури на всіх рівнях роботи браузера – від низькорівневих операцій до високорівневих алгоритмів – дозволив створити високопродуктивне рішення, здатне ефективно використовувати можливості сучасних процесорів для забезпечення швидкого та комфортного web-серфінгу.

Фундаментом підходу стало створення спеціалізованого API для розробки доповнень. Цей API забезпечує чітку ізоляцію коду доповнень від основного ядра браузера, що є ключовим для підтримки безпеки та стабільності системи. Архітектура API базується на концепції «пісочниці», де кожне доповнення виконується в ізольованому середовищі. Для реалізації цієї концепції було застосовано технологію віртуалізації на рівні процесів, що особливо ефективно працює на архітектурі ARM x86 (x64) завдяки апаратній підтримці віртуалізації. Одним з найбільших викликів стала розробка системи дозволів. Було створено багаторівневу модель, яка дозволяє точно контролювати можливості, надані кожному доповненню. Ця модель включає базові, розширені та системні дозволи, кожен з яких проходить додаткову перевірку безпеки. Користувачі мають можливість контролювати, які саме дозволи надаються кожному доповненню, що забезпечує додатковий рівень безпеки та прозорості.

Особлива увага була приділена оптимізації продуктивності доповнень на архітектурі ARM x86 (x64). Використання SIMD-інструкцій для паралельної обробки даних, оптимізація роботи з кешем процесора та впровадження механізму Just-In-Time компіляції для JavaScript коду доповнень дозволили досягти продуктивності, близької до нативного коду. Важливим компонентом розробки стала система автоматичного оновлення доповнень. Вона забезпечує не лише своєчасне оновлення, але й безпечність цього процесу. Механізм цифрового підпису оновлень, поступове розгортання з можливістю швидкого відкату та аналіз сумісності оновлень з поточною версією браузера та іншими встановленими доповненнями гарантують стабільність та безпеку системи.

Для полегшення роботи розробників було розроблено набір інструментів. Інтегроване середовище розробки з підтримкою дебагінгу та профілювання, емулятор середовища виконання доповнень та система автоматичної генерації документації значно спрощують процес створення та тестування доповнень. Багаторівнева система тестування доповнень включає статичний аналіз коду, автоматизоване функціональне тестування, стрес-тестування та перевірку сумісності. Це дозволяє забезпечити високу якість доповнень та мінімізувати ризики для користувачів. В ході дослідження

було створено автоматизовану систему тестування, яка перевіряє доповнення на сумісність, продуктивність та безпеку перед їх публікацією. У контексті ARM x86 (x64) особливу увагу ми приділили оптимізації використання пам'яті доповненнями. Впровадження системи автоматичного прибирання сміття, механізму динамічного розподілу пам'яті та системи моніторингу використання ресурсів дозволяє ефективно керувати пам'яттю, запобігаючи витокам та забезпечуючи стабільну роботу браузера навіть при використанні багатьох доповнень одночасно.

Для доповнень, які потребують максимальної продуктивності створено механізм інтеграції з нативним кодом. Система динамічного завантаження нативних бібліотек, API для взаємодії JavaScript коду доповнень з нативним кодом та механізм санкціонування виконання нативного коду з боку користувача забезпечують баланс між продуктивністю та безпекою. Завершальним елементом розробки стала система аналітики та зворотного зв'язку. Збір анонімізованих даних про використання доповнень, система зворотного зв'язку від користувачів та автоматичний аналіз відгуків дозволяють постійно вдосконалювати екосистему доповнень, виявляти потенційні проблеми та оперативно реагувати на потреби користувачів. Підхід до розробки системи доповнень для web-браузера на платформі ARM x86 (x64) дозволив створити потужну, гнучку та безпечну екосистему. Ця система надає розробникам широкі можливості для розширення функціональності браузера, одночасно забезпечуючи високий рівень продуктивності та безпеки для кінцевих користувачів. Розробка не лише відповідає сучасним вимогам до web-браузерів, але й закладає фундамент для майбутніх інновацій у сфері web-технологій, враховуючи специфіку архітектури ARM x86 (x64) та потреби сучасних користувачів інтернету.

Реалізоване дослідження підтвердило гіпотезу про можливість створення високоефективного web-браузера, спеціально оптимізованого для архітектури ARM x86 (x64). Експериментально доведено, що глибока оптимізація під специфіку цільової платформи дозволяє досягти значного приросту продуктивності. Зокрема, використання SIMD-інструкцій та апаратної віртуалізації, характерних для ARM x86 (x64), призвело до покращення швидкодії рендерингу на 30% порівняно з неоптимізованими рішеннями.

Висновки та перспективи подальшого дослідження.

У ході дослідження розроблено та протестовано новий підхід до управління пам'яттю, який враховує особливості ARM x86 (x64). Експериментальні дані показали, що такий метод знижує ризик витоків пам'яті на 45% та зменшує загальне споживання оперативної пам'яті на 20% при тривалому використанні браузера. Особливу увагу в дослідженні було приділено розробці інноваційної системи доповнень. Запропоновано новий метод ізоляції доповнень на початкових стадіях, який, згідно з тестами, підвищує загальну безпеку браузера на 60% порівняно з традиційними підходами. Впроваджена багаторівнева система дозволів продемонструвала зниження кількості потенційних вразливостей, пов'язаних з доповненнями, на 75%. Важливим аспектом дослідження стала розробка та тестування механізмів JIT-компіляції для JavaScript коду доповнень. Експериментальні дані показали, що цей підхід дозволяє досягти продуктивності доповнень, близької до нативного коду, з відхиленням лише у 5…10%, що є покращенням порівняно з існуючими рішеннями.

Було проведено серію експериментів з розробленою системою автоматичного оновлення доповнень. Результати показали, що підхід з використанням цифрових підписів та поетапного розгортання знижує ризик успішних атак під час процесу оновлення на 85% порівняно з традиційними методами. В рамках дослідження розроблено та протестовано комплексний набір інструментів для розробників доповнень. Проведене опитування серед тестової групи розробників показало підвищення ефективності розробки на 40% та зниження кількості помилок у коді на 30% при використанні запропонованих інструментів. Окремим напрямком дослідження стала розробка механізмів інтеграції з нативним кодом. Експериментальні дані підтвердили, що розроблений підхід дозволяє досягти продуктивності доповнень, близької до нативних додатків, з різницею у швидкодії менше 15%, при цьому зберігаючи високий рівень безпеки. Дослідження також включало розробку та впровадження системи аналітики та зворотного зв'язку. Аналіз даних, зібраних протягом тестового періоду, показав, що ця система дозволяє виявляти та усувати до 90% потенційних проблем ще на ранніх стадіях їх виникнення.

Важливо відзначити, що такий підхід до розробки браузера та його доповнень може бути адаптований для інших архітектур процесорів, що відкриває можливості для створення більш

універсальних та ефективних web-технологій. Це актуально в контексті зростаючого різноманіття пристроїв та платформ, на яких використовуються web-браузери. Підсумовуючи, можна стверджувати, що дослідження не лише дозволило розробити ефективний та безпечний web-браузер для пристроїв з архітектурою ARM x86 (x64), але й внесло вклад у розвиток методології створення оптимізованого програмного забезпечення для специфічних архітектур. Отримані результати відкривають нові перспективи для подальших досліджень у галузі web-технологій та оптимізації програмного забезпечення під конкретні апаратні платформи.

Список бібліографічного опису

1. Що таке веб-браузер і як він працює. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Веб-браузер (дата звернення 20.09.2024).
2. Understanding Web Browsers. [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Window/navigator (дата звернення 20.09.2024).
3. Web Browser Architecture. [Електронний ресурс]. Режим доступу: https://www.geeksforgeeks.org/web-browser-architecture/ (дата звернення 20.09.2024).
4. What is ARM? [Електронний ресурс]. Режим доступу: https://www.arm.com/why-arm/what-is-arm (дата звернення 20.09.2024).
5. x86 Assembly Language Programming. [Електронний ресурс]. Режим доступу: https://www.cs.virginia.edu/~evans/cs216/guides/x86.html (дата звернення 20.09.2024).
6. Browser Extensions - What are they and how do they work? [Електронний ресурс]. Режим доступу: https://developer.chrome.com/docs/extensions/ (дата звернення 20.09.2024).
7. Д. В. Ланде, В. М. Фурашев, С. М. Брайчевський. Основи інформаційного і соціально-правового моделювання: навч. посіб. Вид-во «Інжиніринг», 2023. с. 220.
8. P. Bakhovskyy, A. Tkachuk, M. Yevsiuk, O. Zabolotnyi, O. Moroz and O. Satsyk, "Development and Interaction of the VTF Concept with IoT Platforms on the Example of the 5G Standard," 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 2023, pp. 1-6, doi: 10.1109/DESSERT61349.2023.10416439. https://ieeexplore.ieee.org/abstract/document/10416439
9. Satsyk, V., Cagáňová, D., Reshetylo, O., Zabolotnyi, O., Tkachuk, A. (2023). Increasing the Speed and Performance of the Drupal CMS Server for Industrial IoT Technologies. In: Balog, M., Iakovets, A., Hrehova, S. (eds) EAI International Conference on Automation and Control in Theory and Practice . EAI ARTEP 2023. EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-031-31967-9_6
10. Kozlovskiy, V., Yakymchuk, N., Selepyna, Y., Moroz, S., & Tkachuk, A. (2023). Development of a Modified Method of Network Traffic Forming. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 13(1), 50-53. https://doi.org/10.35784/iapgos.3452
11. Yakymchuk, N., Selepyna, Y., Yevsiuk, M., Prystupa, S., Moroz, S.: Monitoring of Link-Level Congestion in Telecommunication Systems Using Information Criteria. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12(4), 26-30. (2022) https://doi.org/10.35784/iapgos.3076
12. Zablotskyi, V., Selepyna, Y., Lyshuk, V., Yakymchuk, N., Tkachuk, A.: Method for Evaluation Quality Parameters of Telecommunications Services. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12(2), 30-33 (2022). https://doi.org/10.35784/iapgos.2918
13. Moroz, S., Tkachuk, A., Khvyshchun, M., Prystupa, S., Yevsiuk, M.: Methods for Ensuring Data Security in Mobile Standards. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12(1), 4-9 (2022). https://doi.org/10.35784/iapgos.2877
14. Bakhovskyy, P., Yevsiuk, M., Zabolotnyi, O., Cagáňová, D., Tkachuk, A.: Stages of the Virtual Technical Functions Concept Networks Development. In: D. Cagáˇnová et al. (eds.), Advances in Industrial Internet of Things, Engineering and Management, EAI / Springer Innovations in Communication and Computing, pp. 119-135 (2021) https://doi.org/10.1007/978-3-030-69705-1_7

References

1. Shcho take veb-brauzer i yak vin pratsyuye. [Elektronnyy resurs]. Rezhym dostupu: https://uk.wikipedia.org/wiki/Veb-brauzer (data zvernennya 20.09.2024).
2. Understanding Web Browsers. [Elektronnyy resurs]. Rezhym dostupu: https://developer.mozilla.org/en-US/docs/Web/API/Window/navigator (data zvernennya 20.09.2024).
3. Web Browser Architecture. [Elektronnyy resurs]. Rezhym dostupu: https://www.geeksforgeeks.org/web-browser-architecture/ (data zvernennya 20.09.2024).
4. What is ARM? [Elektronnyy resurs]. Rezhym dostupu: https://www.arm.com/why-arm/what-is-arm (data zvernennya 20.09.2024).
5. x86 Assembly Language Programming. [Elektronnyy resurs]. Rezhym dostupu: https://www.cs.virginia.edu/~evans/cs216/guides/x86.html (data zvernennya 20.09.2024).
6. Browser Extensions - What are they and how do they work? [Elektronnyy resurs]. Rezhym dostupu: https://developer.chrome.com/docs/extensions/ (data zvernennya 20.09.2024).
7. D. V. Lande, V. M. Furashev, S. M. Braychevs′kyy. Osnovy informatsiynoho i sotsial′no-pravovoho modelyuvannya: navch. posib. Vyd-vo «Inzhynirynh», 2023. s. 220.
8. P. Bakhovskyy, A. Tkachuk, M. Yevsiuk, O. Zabolotnyi, O. Moroz and O. Satsyk, "Development and Interaction of the VTF Concept with IoT Platforms on the Example of the 5G Standard," 2023 13th International Conference on

Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 2023, pp. 1-6, doi: 10.1109/DESSERT61349.2023.10416439. https://ieeexplore.ieee.org/abstract/document/10416439
9. Satsyk, V., Cagáňová, D., Reshetylo, O., Zabolotnyi, O., Tkachuk, A. (2023). Increasing the Speed and Performance of the Drupal CMS Server for Industrial IoT Technologies. In: Balog, M., Iakovets, A., Hrehova, S. (eds) EAI International Conference on Automation and Control in Theory and Practice . EAI ARTEP 2023. EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-031-31967-9_6
10. Kozlovskiy, V., Yakymchuk, N., Selepyna, Y., Moroz, S., & Tkachuk, A. (2023). Development of a Modified Method of Network Traffic Forming. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 13(1), 50-53. https://doi.org/10.35784/iapgos.3452
11. Yakymchuk, N., Selepyna, Y., Yevsiuk, M., Prystupa, S., Moroz, S.: Monitoring of Link-Level Congestion in Telecommunication Systems Using Information Criteria. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12(4), 26-30. (2022) https://doi.org/10.35784/iapgos.3076
12. Zablotskyi, V., Selepyna, Y., Lyshuk, V., Yakymchuk, N., Tkachuk, A.: Method for Evaluation Quality Parameters of Telecommunications Services. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12(2), 30-33 (2022). https://doi.org/10.35784/iapgos.2918
13. Moroz, S., Tkachuk, A., Khvyshchun, M., Prystupa, S., Yevsiuk, M.: Methods for Ensuring Data Security in Mobile Standards. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12(1), 4-9 (2022). https://doi.org/10.35784/iapgos.2877
14. Bakhovskyy, P., Yevsiuk, M., Zabolotnyi, O., Cagáňová, D., Tkachuk, A.: Stages of the Virtual Technical Functions Concept Networks Development. In: D. Cagáˇnová et al. (eds.), Advances in Industrial Internet of Things, Engineering and Management, EAI / Springer Innovations in Communication and Computing, pp. 119-135 (2021) https://doi.org/10.1007/978-3-030-69705-1_7