ISSN 2219-9365

https://doi.org/10.31891/2219-9365-2024-78-31
УДК 004.4

МЕЛЬНИК Вікторія
Хмельницький національний університет
https://orcid.org/0009-0009-7668-4318
gutsalykvm@gmail.com
ЛИГУН Олексій
Хмельницький національний університет
https://orcid.org/0009-0004-5727-5096
oleksii.lyhun@gmail.com
ВОЗНИЙ Кирило
Хмельницький національний університет
https://orcid.org/0009-0007-2545-565X
k.vozniy@gmail.com
ГУРАЛЬНИК Олександр
Хмельницький національний університет
https://orcid.org/0009-0009-1175-8726
gurualexua@gmail.com
МАРТИНЮК Дмитро
Хмельницький національний університет
https://orcid.org/0009-0002-3524-872X
martiniyuk.dim14@gmail.com

ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ В СИСТЕМАХ АВТОМАТИЗАЦІЇ БУДІВЕЛЬ

В роботі проаналізовано будівлі, які є контейнерами безлічі різних видів людської діяльності, що підтримуються різними видами технологій. Тип і корисність будівлі визначають її унікальні потреби в безпеці. Відсутність належної сумісності та стандартів безпечної архітектури створює потенційні небезпеки та загрози для контролю, особливо для критично важливих для безпеки об'єктів. З іншого боку, будівлі неоднорідні. Різні будівлі призначені для різних цілей, тому мають різні вимоги. Запропоноване рішення є надійним завдяки мінімальній функціональності мікроядра в просторі ядра та модульній архітектурі. При цьому має зворотну сумісність, оскільки застарілі програми можуть бути розміщені у віртуальному середовищі, а архітектуру проксі-сервера захищає протокол мережевого зв'язку систем автоматизації будівель за допомогою мережевого тунелювання. Для оцінки запропонованого рішення були реалізовані різні сценарії автоматизації будівель, такі як сценарій лабораторного контролю температури, сценарій управління ключами та відповідні атаки. Ці експерименти показують переваги безпеки запропонованого рішення та відчутну ефективність.

Представлений підхід віртуалізації використовує формально перевірений як мікровізор для розміщення застарілих систем автоматизації будівель. Цей підхід ґрунтується на полегшенні адаптації, одночасно збалансовуючи компроміси між витратами та безпекою. Крім того, він використовує формально перевірене мікроядро для його унікальних переваг доведеної гарантії безпеки та прогресу в технології віртуалізації. Система може розвиватися з часом, оскільки програмні продукти змінюються. Розроблена безпечна обчислювальна платформа для системи аутентифікації будівель наступного покоління за технологією кіберфізичних систем.

Напрямами подальших досліджень є удосконалення в підході, що базується на модифікаціях в програмному забезпеченні засобів IoT. Ці удосконалення стосуватимуться врахування протоколів взаємодії засобів та інтерфейсів операційних систем.

Ключові слова: безпека, система автоматизації будівель, технології IoT.

MELNYK Viktoriia, LYHUN Oleksii, VOZNYI Kyrylo,
HURALNYK Oleksandr, MARTINIUK Dmytro
Khmelnitskyi National University

ENSURING SAFETY IN BUILDING AUTOMATION SYSTEMS

The work analyzes buildings that are containers of many different types of human activity supported by different types of technologies. A building's type and utility determine its unique security needs. Lack of proper interoperability and security architecture standards creates potential hazards and threats to control, especially for security-critical facilities. On the other hand, buildings are heterogeneous. Different buildings are designed for different purposes and therefore have different requirements. The proposed solution is robust due to minimal kernel-space microkernel functionality and modular architecture. At the same time, it has backward compatibility, since outdated programs can be placed in a virtual environment, and the proxy server architecture is protected by the network communication protocol of building automation systems using network tunneling. To evaluate the proposed solution, various building automation scenarios such as laboratory temperature control scenario, key management scenario and corresponding attacks were implemented. These experiments show the security advantages of the proposed solution and the tangible efficiency.

The presented virtualization approach uses a formally proven micro-visor for hosting legacy building automation systems. This approach is based on facilitating adaptation while balancing cost and security trade-offs. In addition, it uses a formally proven microkernel for its unique benefits of proven security assurance and advances in virtualization technology. The system may evolve over time as software products change. A secure computing platform for the next-generation building authentication system based on cyber-physical systems technology has been developed.

Directions for further research are improvements in the approach based on modifications in the software of IoT devices. These improvements will be related to taking into account the protocols of interaction of tools and interfaces of operating systems.
Keywords: security, building automation system, IoT technologies.

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Будівлі є контейнерами безлічі різних видів людської діяльності, що підтримуються різними видами технологій. Тип і корисність будівлі визначають її унікальні потреби в безпеці [1, 2]. Відсутність належної сумісності та стандартів безпечної архітектури створює потенційні небезпеки [3] та загрози для контролю, особливо для критично важливих для безпеки об'єктів. З іншого боку, будівлі неоднорідні [4]. Різні будівлі призначені для різних цілей, тому мають різні вимоги. Наприклад, система автоматизації стадіону може бути зосереджена на тому, як керувати освітленням, температурою та посиленою вентиляцією для підтримки низької концентрації вуглекислого газу, тоді як основною проблемою для біологічної ізоляції або хімічного заводу може бути мінімізація повітрообміну між різними зонами для зменшення ризику перехресного забруднення.

Замість того, щоб покладатися на традиційні підходи до безпеки [5-7], такі як контроль периметра та постійний моніторинг і виправлення, вважається, що вбудовані контролери повинні прийняти радикально новий архітектурний підхід до безпеки та захисту [8], щоб основні примітиви, що підтримують ці властивості, могли бути вбудовані, а не додані пізніше. У зв'язку з вродженою динамікою навколишнього середовища будівлі контролери повинні бути в змозі гарантувати надійну безпеку в агресивних умовах, навіть якщо деякі з них не довіряють або скомпрометовані [9]. Дуже важливим першим кроком є дослідження безлічі проблем безпеки та захисту, які можуть виникнути, і належне їх моделювання, щоб кожен вбудований контролер у навколишнє середовище мав чітко визначені вимоги безпеки/безпеки, яких він повинен дотримуватися [10]. Ґрунтуючись на аналізі сценаріїв, можна побудувати модулі конкретної системи автоматизації будівлі та визначити обмеження безпеки чи захисту, які повинні бути забезпечені різними рівнями системи [11], що працює на цих вбудованих контролерах. Нарешті, можна створити багаторівневу архітектуру для вбудованих платформ, щоб програми могли використовувати ці властивості для досягнення безпеки та захисту саме для своїх контекстів [12].

Тому, проєктуватимемо рішення виходячи з того, що операційна система (ОС) на основі мікроядра є ядром платформи вбудованих контролерів, завдяки перевагам мікроядра.

Аналіз відомих рішень щодо забезпечення безпеки автоматизованих будівель

Ядро з базовою апаратною підтримкою забезпечує такі функціональні можливості [13]: ізоляцію процесу з метою забезпечити цілісність логіки керування; планування з обмеженням у реальному часі з метою забезпечити оперативність та доступність; міжпроцесну комунікацію та синхронізацію з метою забезпечити цілісність та автентичність потоку даних; мережевий зв'язок з метою забезпечити захищений зв'язок та конфіденційність. Крім завдань контролю повітряного потоку і тиску, існують інші завдання управління, які співіснують на одних і тих же контролерах в зоні дослідження будівель. Розглянемо завдання контролю безпеки [14] для забезпечення дотримання політик контролю доступу. Кожна кімната самозачиняються і дозволяє доступ лише авторизованим мешканцям. Система контролю доступу в кожній зоні управляється локальним контролером безпеки і централізовано управляється сервером фізичного доступу. У режимі роботи входити та виходити, коли доступ буде заборонено не можна, доки небезпека не буде знята.

Завдання на керування режимами для координації різних режимів роботи будівлі відбувається через управління режимами та координується глобальним контролером і кімнатними контролерами [15]. Для зони передбачено режими: режим без виділення, який вказує на етап ініціалізації зони до того, як фізичне середовище досягне потрібного рівня, наприклад, температура, тиск повітря; режим роботи, який свідчить про те, що всі лабораторії готові до використання; режим пожежі, який сигналізує про пожежу; режим дезактивації, який спрацьовує, коли відбувається процедурне порушення або фізичне середовище падає до небажаного рівня нижче порогового значення [16]. Управління кожною зоною здійснюється відносно незалежно за допомогою глобального контролера, яким керують система управління будівлею і система енергетичного менеджменту [17]. У зоні контроль доступу для всіх дверей здійснюється контролером безпеки, яким керує сервер фізичного доступу, тоді як керування блокуванням контролюється одним контролером блокування з магнітними приводами блокування та датчиками положення дверей, які застосовуються до кожних дверей. У кожній кімнаті є локальний контролер для всіх інших завдань керування з відповідними датчиками та виконавчими механізмами. Ці контролери обмінюються даними один з одним через протокол [18].

На відміну від сценарію для систем, що потребують ізоляції, будівлі університетських кампусів, такі як бібліотеки, гуртожитки, студентські центри тощо, призначені для забезпечення легкого та комфортного публічного доступу для великих груп та проведення різних навчальних та дослідницьких заходів. У

будівлях кампусу ізоляція приміщень та обмеження несанкціонованого доступу не викликають великого занепокоєння. Викладачі, співробітники та студенти проводять значну частину свого неспання в університетських будівлях. Підвищення рівня безпеки та комфорту мешканців, а також енергоефективність будівель кампусу є одними з головних пріоритетів системи автоматизації будівель (САБ).

Основною метою управління системами опалення, вентиляції та кондиціонування повітря для будівлі кампусу САБ є ефективне збільшення повітрообміну [19]. На відміну від сценарію, де припливне повітря повністю надходить із зовнішнього повітря через фільтри, з міркувань безпеки для будівлі кампусу енергоефективніше максимально повторно використовувати циркулююче повітря. Зазвичай свіже зовнішнє повітря, що надходить у будівлю, має бути оброблене для видалення зайвої вологи та охолоджене/нагріття до заданої температури перед подачею в будівлю. Охолодження, нагрівання та осушення повітря становить значну частину загальних витрат на експлуатацію систем опалення, вентиляції та кондиціонування повітря, особливо в районах, де погода, як правило, спекотна та волога. Таким чином, зменшення потреби в обробці зовнішнього повітря шляхом очищення та повторного використання внутрішнього повітря є важливим методом енергозбереження [20]. Повторне використання рециркуляційного повітря підвищує рівень вуглекислого газу у приміщенні. Вуглекислий газ є природним компонентом повітря. Підвищена концентрація може викликати сонливість, млявість і синдром хворої будівлі. Тому підтримка комфортного рівня концентрації в будівлях кампусу, де великі групи людей постійно видихають вуглекислий газ, має вирішальне значення [21].

Для зон з високою щільністю населення ще однією проблемою є екстрена евакуація під час пожежі та задимлення. Підсистема управління системами опалення, вентиляції та кондиціонування повітря САБ відповідає за виявлення таких інцидентів, зменшення потенційної шкоди та полегшення процедур евакуації для громадської безпеки за допомогою таких заходів, як відкриття або відмикання дверей пожежних виходів, увімкнення сигналізації пожежної сигналізації та зменшення подачі свіжого повітря в зону, що горить [22].

Щоденна послідовність роботи системи опалення, вентиляції та кондиціонування повітря наступна. Спочатку, виходячи з зовнішньої температури та вологості, керівник будівлі запускає роботу в режимі обігріву або охолодження з потрібним рівнем вологості. Коли розпочнеться робочий день, в запланований час контролер переведе систему в зайнятий режим. Інші контролери періодично перевіряють за допомогою контролеру зміну режиму. Контролер зворотного повітря стежить за вологістю, температурою, рівнем вуглекислого газу, а також за станом пожежної тривоги. Він регулює заслінку зворотного повітря і відкриває/закриває її за бажанням керівника. У зайнятому режимі контролер зовнішнього повітря постійно регулює зовнішні заслінки, посилаючись на дані датчика від контролера зворотного повітря [23]. У зайнятому режимі контролер обробляє подачу повітря. Клапан регулювання охолодженої води змійовика охолодження підтримується на певному рівні на основі зворотного зв'язку датчика та заданого значення режиму для зниження вологості з повітря, що всмоктується. Швидкість припливного вентилятора автоматично регулюється за допомогою датчика перепаду тиску зворотного зв'язку між воздуховодом і коридором. Також є повітряний фільтр, який допомагає контролювати пил. Між двома сторонами повітряного фільтра встановлено статичний датчик, який визначає, коли фільтр засмічений настільки, що його можна замінити. Двоповерхові контролери заслінок розгорнуті в нижній частині повітропроводу. Контролери підлогових демпферів визначають різницю тиску між повітропроводом і підлоговим коридором і використовують зворотний зв'язок для управління вхідними направляючими лопатками і заслінкою. За це відповідає контролер димовидалення для відведення диму з будівлі. У зайнятому режимі контролер відведення диму зазвичай переводиться в сплячий режим.

Під час пожежі та задимлення спрацьовує система пожежної сигналізації та сповіщається контролер. Це змінює задане значення режиму з зайнятого режиму на режим відведення диму [49-54]. Потім зміна режиму поширюється на всі контролери системи опалення, вентиляції та кондиціонування повітря. Контролер зворотного повітря повністю закриє заслінку для запобігання рециркуляції диму в будівлі [55]. Припливний вентилятор у вентиляційній установці працюватиме на повну потужність. Мета полягає в тому, щоб зменшити поширення диму на всю будівлю, забезпечуючи при цьому достатню кількість свіжого повітря, насиченого киснем, для безпеки персоналу [24]. У нижній частині воздуховода контролери підлогових заслінок відкриють заслінку для відведення подачі повітря. Причина цього полягає в тому, щоб зменшити постачання кисню в офісних приміщеннях, де найімовірніше станеться пожежа, і збільшити концентрацію кисню в коридорах для полегшення евакуації. Тим часом контролер відведення диму відкриє заслінку і запустить витяжний вентилятор, щоб максимально витягнути дим, поки пожежна сигналізація не буде вимкнена або ручним скиданням.

Аналіз цих двох сценаріїв показує, що САБ це дуже складні системи [15-24], орієнтовані на дані. Існують різні аналогові, цифрові та мережеві пристрої з даними з різних датчиків. Сумісність підсистем САБ ускладнює проектування інтегрованої САБ. Різні компоненти взаємодіють один з одним через мережу управління, а зміни у фізичному середовищі також опосередковано впливають на прийняття рішень іншими підсистемами [25]. Наприклад, зміна температури може спричинити коливання тиску повітря, тому

вимірювання перепаду тиску також має враховувати температуру в приміщенні. Часто стан і успішність завдання управління, наприклад, стан повітряного фільтра або відкриття заслінки, опосередковано позначаються фізичним станом, зафіксованим датчиками. Крім того, логіка різних контрольних завдань змінюється відповідно до інституційної політики в різних ситуаціях. Помилки в логіці управління і неправильні припущення в САБ можуть спричинити серйозну загрозу безпеці. Наприклад, у механізмах виявлення дезактивації можуть бути реалізовані дві взаємосуперечливі задачі. Якщо впровадити без ретельного обмірковування всіх можливих ситуацій у сценарії, контроль САБ може призвести до того, що забруднення ніколи не буде виявлено належним чином.

Таким чином, важливою особливістю систем автоматизаці будівель є правильне проєктування, набори апаратного забезпечення та безпосередньо забезпечення правильної організації функціонування таких систем.

Технології IoT в системах автоматизації будівель

Тенденція до більш інтелектуального управління в САБ вимагає інтеграції даних і нових можливостей в пристроях управління. Для того, щоб задовольнити цю вимогу, сучасний САБ вимагає більшої обчислювальної потужності. З розвитком апаратного забезпечення багато постачальників контролерів автоматизації поступово переходять на архітектуру як основну обчислювальну платформу поточних і майбутніх контролерів будівель. У відповідь на ці розробки апаратні технології САБ кардинально змінилися від 8-розрядних до 64-розрядних процесорів, від одноядерних до багатоядерних. Поряд з логікою управління програмним забезпеченням, хмарні клієнти, веб-сервер, аналітика часто розгортаються на одних і тих же вбудованих контролерах, що підвищує складність системи і робить її схильною до потенційних вразливостей і експлойтів.

Незважаючи на те, що в дослідженні безпеки є багато перспективних пропозицій [4-8], дуже рідко можна побачити, як ці рішення використовуються в реальних продуктах. Завдання модернізації існуючої САБ для задоволення вимог змішаної системи критичності є двояким. По-перше, безпосереднє застосування безпеки в застарілій системі є неможливим через апаратні обмеження та відсутність міркувань безпеки при початковому проектуванні. За останні 10 років було розгорнуто багато існуючих контролерів будівель, наприклад, програмовані логічні контролери. Ці контролери часто не розрізняють режим користувача та режим привілеїв. Крім того, в цих старих моделях різні завдання управління вбудовані в один і той же простір пам'яті без операційної системи, щоб допомогти ізолювати один одного. З огляду на їх вік, модернізувати такі пристрої по суті неможливо. Викриття цих пристроїв в Інтернеті пов'язане з великими ризиками. Повна заміна застарілої системи є непомірно дорогою. Перенесення існуючого програмного забезпечення є дорогим і трудомістким процесом. Дослідження показує, що в бізнесі перенесення існуючого програмного забезпечення на інші платформи, як правило, коштує 30% від загальної вартості розробки нового продукту. Багато постачальників мають власні апаратні платформи з індивідуальними операційними системами для підтримки свого програмного забезпечення. Наприклад, для відомого рішення автоматизації використовують вбудований контролер IoT, для якого розроблено своє програмне забезпечення на відносно новому процесорі ARM і операційну систему реального часу, яка використовується протягом останнього десятиліття. Це рішення, ймовірно, є одним з найбільш схожих комерційних застосувань у порівнянні із запропонованим прототипом ОС з підвищеною безпекою з точки зору апаратної платформи та архітектури ОС. Незважаючи на схожість, нетривіально перенести існуючу кодову базу на запропоновану ОС з підвищеною безпекою, не кажучи вже про перенесення багатьох старих програм на інші рішення. З іншого боку, новим ОС часто не вистачає підтримки драйверів пристроїв від сторонніх постачальників, що ще більше ускладнює процес прийняття.

Ці виклики підтверджуються практикою. Розробники САБ підтримують споживчі та промислові рішення для автоматизації будівель з технологіями IoT. Прагнучи інтегрувати продукти управління з хмарними платформами для забезпечення «розумних» підключених активів і розширеної аналітики, ядром цього рішення є розумний шлюз IoT. Шлюз відповідає за збір даних пристрою та зв'язок із хмарними платформами. Отже, безпека шлюзу має вирішальне значення, оскільки компрометація шлюзу дозволить атакам отримати доступ до хмарних активів і керувати виконавчими пристроями. Тому, краще побудувати такий шлюз за допомогою вищезгаданої платформи мікроядра безпеки. В осяжному майбутньому нереально відмовитися від існуючої роботи на користь кращої безпеки. Отже, будь-яке практичне рішення має бути ненав'язливим і сприяти цьому переходу в поступовому поступовому процесі. Відмовлятися від застарілих стеків програмного забезпечення для бізнесу непомірно дорого. Неможливо очікувати, що компанії створюватимуть застосунки з нуля на користь безпеки.

Одним із рішень для конвергенції застарілого програмного забезпечення та нової платформи є віртуалізація. Віртуалізація відноситься до технологій, призначених для забезпечення віртуального середовища виконання для розміщення програмного забезпечення, яке традиційно працює на фізичних комп'ютерах. Віртуалізація була ключовим фактором технології хмарних обчислень. Це дозволяє різним операційним системам ефективно використовувати однакові апаратні ресурси та зберігати незалежність.

Цей метод розділяє фізичні ресурси на кілька часток, що робить застосунки більш доступними, зменшує кількість необхідних ресурсів, забезпечує краще управління і, таким чином, скорочує витрати. Найголовніше, що програми не повинні залежати від конкретної ОС.

З міркувань вартості та безпеки переважає віртуалізація на вбудованій системі на кристалі. Тим більше, що з передовими апаратними асистентами вплив на зниження продуктивності є мінімальним. Основна ідея віртуалізації полягає в тому, щоб абстрагувати апаратне забезпечення, наприклад, процесор, пам'ять, мережу і т.д., в ізольовані середовища виконання, щоб дозволити кожному окремому середовищу працювати так, ніби на приватному комп'ютері. Віртуалізація приносить бажану користь вбудованим пристроям. Віртуалізація дозволяє консолідувати кілька різнорідних хостів з багатими функціональними можливостями на одному процесорі. Використання вбудованої віртуалізації не тільки знижує витрати, але й забезпечує можливість розміщення застарілих програм. Найсильнішою мотивацією для вбудованої віртуалізації є те, що вона підвищує безпеку САБ шляхом ізоляції програмного забезпечення в середовищах виділення. Ця ідея спеціально підходить для систем змішаної критичності в САБ. Якщо ізоляція між віртуальними середовищами може бути гарантована, а політика зв'язку може суворо застосовуватися відповідно до специфікації, то критичні процеси в САБ матимуть власний виділений віртуальний процесор і власний виділений віртуальний простір пам'яті, навіть якщо вони розміщені спільно в одних і тих же фізичних контролерах. Вплив потенційних порушень можна мінімізувати, запустивши ці контрольні завдання у віртуальній машині. Ще краще те, що застаріле програмне забезпечення може вимагати дуже мінімальних модифікацій. Враховуючи, що монолітні ядра за своєю суттю неможливо захистити без шкоди для зручності використання та сумісності, підхід мікроядра представляє альтернативне рішення. Для того, щоб зробити припущення про безпеку обґрунтованим, базовий гіпервізор повинен бути надійним, функціонально правильним без невизначеної поведінки і, бажано, мати якомога менший розмір. З цих причин формально верифіковане мікроядро seL4 є відмінним рішенням для реалізації гіпервізора на основі мікроядра, так званого мікровізора.

Мікроядро — це мінімальна системна база для забезпечення апаратної абстракції для побудови різних системних служб. Гіпервізор призначений виключно для управління базовим апаратним забезпеченням для мультиплексування ресурсів між різними віртуальними середовищами виконання, яке включає як служби користувацького призначення, так і код ядра. Завдяки схожості функціоналу та структури, мікроядро може бути використане для реалізації гіпервізора. Сімейство мікроядер L4 було побудовано як мікровізор для підтримки як пара-віртуалізації, так і апаратної віртуалізації протягом тривалого часу. Їх реалізації продемонстрували належні результати як гіпервізори в промисловості та наукових колах. Однак жодне з існуючих мікроядер не може забезпечити такий же рівень гарантії в порівнянні з seL4. Розроблено мікровізор на базі seL4 для процесора ARM. Використовуючи апаратну віртуалізацію, випустили бібліотеку для побудови VMM. Тому, мікровізор seL4 використовується як фундаментальна платформа для реалізації прототипу контролера САБ. Мікровізор ізолює завдання управління в різні віртуальні машини, але в домені САБ багато завдань управління також повинні взаємодіяти та співпрацювати разом для виконання повсякденних операцій. Тому, необхідний обов'язковий механізм контролю доступу, який дозволяє розробникам застосовувати тонкі політики комунікації лише щодо тих завдань, які дозволені до співпраці. Модель доступу, заснована на можливостях seL4, забезпечує гнучкий метод прозорого посередництва загальносистемних політик безпеки між віртуальними середовищами. У архітектуруі seL4, заснованому на можливостях, все є об'єктом. Кожен об'єкт ядра представляє частку ресурсів, наприклад, сторінки пам'яті, прив'язані до пристрою області, канали зв'язку тощо. Можливість є непідроблюваним токеном із пов'язаними правами доступу, який посилається на відповідний об'єкт ядра. Кожна можливість може бути призначена для конкретного процесу. Ядро керує токеном можливостей від імені процесів. Коли створюється об'єкт, ядро повертає індекс у таблиці можливостей процесу власника, який може бути використаний процесом для посилання на фактичні об'єкти ядра та виклику його функцій. Ядро захищає цю таблицю можливостей, тому її неможливо підробити.

Для того, щоб два процеси могли взаємодіяти один з одним, то процеси повинні володіти можливостями каналу зв'язку, кінцевої точки IPC. Отже, якщо процеси не володіють такою здатністю, Вони не можуть спілкуватися. По суті, загальносистемні політики безпеки застосовуються у формі розподілу можливостей, які можуть бути попередньо налаштовані статично. Щоб полегшити розробку, seL4 надала предметно-специфічну мову для опису розподілу можливостей систем seL4. Вона зчитує специфікацію і переводить її в низькорівневий код C. Вона використовується для створення та розповсюдження як можливостей кінцевих точок IPC, так і спільних сторінок пам'яті між процесами та віртуальними машинами. Згенерований дистрибутив можливостей вбудовано в перший процес, який також служить завантажувачем процесів під час завантаження. Використовуючи мікровізор seL4 та обов'язковий контроль доступу, що примусово контролюється політикою, можемо налаштувати два середовища виконання, рідний розділ seL4 та віртуальну машину Linux. Рідний розділ - це середовище виконання, яке безпосередньо працює поверх seL4 з доступом до API ядра. Розділ забезпечує надійну ізоляцію та вищу безпеку завдяки невеликому TCB, а також моделі доступу на основі можливостей високої впевненості, що забезпечується безпосередньо ядром

seL4. Застосунки в цьому розділі залежать тільки від тонкого шару коду у вигляді процесу seL4 з підтримкою існуючих системних служб і драйверів пристроїв. Втім, з тієї ж причини, що й вище, поки що не існує сумісних із POSIX системних викликів, а драйвери пристроїв обмежені. Отже, застосунок має бути реалізований унікальним способом. З іншого боку, розділ Linux забезпечує багате середовище виконання для існуючих застарілих або низькокритичних програм. За допомогою апаратного забезпечення немодифіковане ядро Linux працює як процес seL4. Оскільки звичайне ядро Linux не зв'язується з бібліотеками seL4 і виконується у власному просторі пам'яті, воно не знає про базовий seL4, і тому неможливо взаємодіяти з програмами, запущеними в рідному розділі. Застосунки в середовищі Linux такі ж, як якщо б вони працювали в звичайній системі Linux. Вони можуть отримати доступ до стандартних API, системних викликів, бібліотек тощо, і більшість із них можуть виконуватися без перенесення. Крім того, деякі драйвери пристроїв у ядрі Linux також можуть бути повторно використані, щоб полегшити відсутність драйверів пристроїв у seL4.

Примусовий зв'язок між розділом Linux і рідним розділом може бути досягнутий за допомогою завантажуваного модуля ядра. Модулі ядра Linux — це фрагменти двійкового коду, які можна динамічно завантажувати та вивантажувати під час виконання. Модулі ядра Linux запускаються в ядрі Linux простір. Це розширює функціональність ядра. У цій конфігурації можна побудувати модуль ядра зв'язку, який знає модель можливостей seL4 та виклики ядра. Коли застосунку в Linux потрібно встановити зв'язок із рідними службами або процесами, програма може викликати зареєстровані системні виклики, і за допомогою цього модуля ядра можуть надсилатися або отримувати повідомлення або дані з кінцевої точки IPC.

Таким чином, це рішення збалансовує зручність використання та безпеку, розміщуючи як середовища виконання Linux, так і природнє середовище високої надійності для практичних цілей. В спробі полегшити процес міграції з традиційного Linux загального призначення на керовану мікроядром систему високої надійності для контролерів в домені САБ, це рішення дозволяє інкрементну стратегію спочатку на конкретних функціях, дозволяючи при цьому продовжувати створювати решту системи.

Розглянемо реалізацію запропонованого прототипу з використанням мікровізора seL4. Використовуючи мікровізор seL4 з відкритим вихідним кодом, це рішення налаштовує платформу з двома середовищами виконання, одним розділом Linux, реалізованим налаштованою віртуальною машиною Linux, і одним рідним розділом з природніми процесами seL4. Крім того, впроваджено безпечну систему завантажувального ланцюга для захисту цілісності програмного забезпечення. Процедура безпечного завантаження застосовує криптографічні методи в процесі завантаження, які перевіряють цифровий підпис кожного компонента та гарантують, що лише авторизовані компоненти можуть бути завантажені на платформу. Розроблене безпечне середовище виконання з використанням мікровізора seL4 зображене на рис. 1.

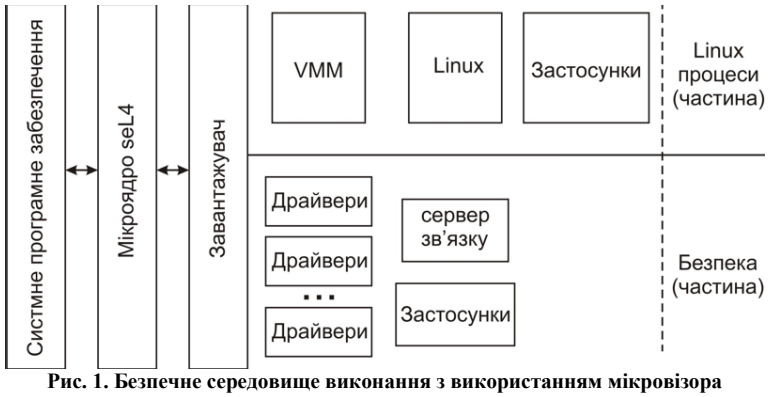

Рис. 1. Безпечне середовище виконання з використанням мікровізора

Для гостьової ОС Linux система запускає налаштований звичайний Linux, який перехресно компілюється для цільової дошки. Ядро Linux та образ побудовані за допомогою вбудованої системи збірки Linux, одного з популярних інструментів, які автоматизують процес. Linux містить модуль ядра, який при завантаженні в систему зареєструється як віртуальний пристрій, який може взаємодіяти з застосунками Linux через системний виклик. Останнім компонентом системи є сервер зв'язку. Сервер зв'язку є рідним процесом seL4. Метою цього процесу є обмін даними між процесами Linux, які виконуються у віртуальному середовищі, та застосунками. Сервер зв'язку також є компонентом. Система налаштовує спільну сторінку пам'яті між віртуальною машиною та сервером зв'язку. Для кожної пари зв'язків між сервером зв'язку та іншими процесами seL4 буде використовуватися виділена кінцева точка IPC. Сервер зв'язку виконує функцію диспетчера. Коли повідомлення отримано або зі спільної пам'яті, або з будь-якої кінцевої точки, повідомлення буде поставлено в чергу та передано відповідному одержувачу.

Незважаючи на те, що рішення віртуалізації добре підходить для розділення застарілих і потенційно вразливих застосунків, воно не заважає зловмисникам фізично втручатися в роботу пристрою або змінювати процес завантаження для завантаження шкідливої ОС. Контрзаходом є надійне завантаження. Безпечне завантаження є критично важливою функцією безпеки, яка гарантує, що з пристрою можна завантажити лише авторизоване програмне забезпечення з цифровим підписом. Безпечне завантаження пов'язує програмне забезпечення з базовим обладнанням, застосовуючи криптографічні методи, такі як цифрові підписи, на кожному етапі процесу завантаження системи, починаючи з моменту ініціалізації обладнання. Оскільки лише двійковий файл із належним підписом може бути перевірений і завантажений у систему, безпечне завантаження гарантує, що лише авторизоване програмне забезпечення може виконуватися на певному обладнанні. Використовуючи це рішення, програмне забезпечення може делегувати довіру через так званий «ланцюжок довіри», починаючи від завантажувача і закінчуючи застосунками, призначеними для користувача. Існують різні стандартні рішення для безпечного завантаження, які пропонують як специфікацію, так і реалізацію, перевірене завантаження тощо. У експерименті плата використовує завантажувач. Реалізовано процедуру безпечного завантаження, яка дозволяє кожному компоненту в системі перевіряти наступний компонент під час процесу завантаження та включати його з перевіреним завантаженням. Після процесу завантаження реалізовано процедуру безпечного завантаження, починаючи в системі seL4. Для цього вставлено відкритий ключ і використано закритий ключ з підписом seL4 ядра і завантажувач, а також додано їх підписи в кінці кожного двійкового файлу. Перед завантаженням завантажувач обчислює хеш кожного двійкового файла та звіряє цифровий підпис зі значенням хешу. Завантажувач продовжить роботу лише в тому випадку, якщо підпис збігається. Потім включено аналогічну процедуру в процес ініціалізації завантажувача, щоб переконатися, що завантажувач перевірить сервер зв'язку та VMM. Нарешті, у VMM додано процедуру перевірки, щоб на платі виконувалися лише підписані ядро Linux та об'єкт дерева пристроїв. Крім того, інтегровано це з перевіреним завантаженням завантажувач. Завантажувач реалізовано з технологією безпечного завантаження, яка дозволяє розробникам вбудовувати криптографічні ключі в завантажувач з метою перевірки. Крім того, перевірене завантаження використовує новий формат файлу, який називається зведеним деревом зображень. Це гнучкий, структурований формат контейнера, який підтримує кілька образів ядра, дерева пристроїв. За допомогою цього формату можна легко додавати підписи та інші метадані, щоб вказати завантажувачу, як завантажити систему. Скомпільовано seL4 та його компоненти в образ, щоб використовувати цей механізм.

Тому, у виробництві багато постачальників підтримують рішення безпечного завантаження, сумісні зі специфікацією безпечного завантаження. Вони часто надають завантажувальні ПЗУ з підтримкою безпечного завантаження, а також додаткове апаратне сховище для програмування ключа шифрування, що не підлягає очищенню. Працюючи з ними, можна вбудовувати криптографічний ключ на платі в майбутньому. Такий ключ може бути використаний для підпису вторинного завантажувача, щоб дозволити лише авторизованому завантажувачу з потрібною конфігурацією бути виконаним на платі для наскрізної гарантії.

Оцінювання отриманих результатів

Щоб оцінити безпеку та практичність цього рішення в домені САБ, прототип прийняв сценарій шлюзу IoT від бізнесу і реалізований у системі. Одним із критично важливих компонентів безпеки в САБ наступного покоління є шлюз IoT. У той час як багато пристроїв САБ замінюються їх останніми аналогами для забезпечення більш «інтелектуального» управління, пристрої все більше і більше покладаються на хмарні обчислювальні служби. Шлюз IoT — це пристрій, який збирає та зберігає дані з предметного поля і забезпечує зв'язок між локальними системами та хмарою. Як периферійний пристрій, який служить точкою входу в об'єднання Інтернету, домашніх і промислових систем автоматизації будівель, шлюз IoT відіграє ключову роль у традиційних САБ та Інтернету послуг. Отже, його безпека нерозривно пов'язана з безпекою та захищеністю об'єкта. Серед усіх функцій шлюзу IoT найбільшою проблемою для бізнесу є захист ключа автентифікації, який використовується для автентифікації, і забезпечення безпечного зв'язку між шлюзом і хмарою. Прагнучи продемонструвати переваги безпеки запропонованого рішення в домені САБ, цей експеримент використовує управління ключами як приклад для оцінки переваг безпеки рішення мікровізора.

У звичайних системах управління ключами покладається на файлову систему і контроль доступу для захисту криптографічних ключів. Під час виконання сервер керування ключами завантажуватиме ключ у свій простір. Сервер керування ключами відповідає за надання криптографічних послуг. Наприклад, шифрування, дешифрування, цифрового підпису, перевірки підпису тощо для авторизованого однорангового процесу. Наприклад хмарного клієнта. За потреби хмарний клієнт зв'яжеться із сервером керування ключами за допомогою сокета локального домену UNIX. Для захисту цих криптографічних ключів від шкідливого доступу, у цій реалізації переміщено керування ключами з частини Linux до розділу seL4. Під час завантаження ключі доступні тільки в розділі seL4 і завантажуються на сервері управління ключами. Хмарний клієнт розміщується в розділі Linux. Коли хмарному клієнту потрібні криптографічні послуги, хмарний клієнт здійснює системний виклик, наприклад, запитує службу ядра або драйверів.

Модуль ядра перевірить привілеї процесу. Якщо запит буде задоволено, модуль ядра перенаправить запит на сервер зв'язку в розділі seL4, викликавши відповідну можливість за допомогою примітиву seL4 IPC. Зрештою, зв'язок передавався на сервер керування ключами, а результат повертався до хмарного клієнта за аналогічним шляхом. З точки зору програми Linux, фактична реалізація є прозорою, отже, потрібно дуже мало змін у вихідному коді.

Щоб оцінити цей підхід, проаналізовано можливі вектори атак у порівнянні зі звичайною реалізацією. Ґрунтуючись на існуючих реальних атаках, модель загроз передбачає атаки двох рівнів. По-перше, атаки віддалено з можливими скомпрометованими процесами. По-друге, атака з фізичним доступом пристрою. Завдяки гарантії функціональної коректності, що надається формальною верифікацією seL4 передбачаєтья, що мікровізор має високу надійність і не містить вразливостей програмного забезпечення. Враховуючи, що шлюз IoT є периферійним пристроєм у звичайній системі, якщо в одній із програм, запущених у розділі Linux, є вразливості, зловмисники можуть отримати віддалений доступ, використовуючи такі слабкі місця. Після того, як зловмисники отримають доступ до пристрою, зловмисники зможуть отримати доступ до криптографічних ключів безпосередньо з файлової системи, якщо дозвіл на файл налаштовано неправильно або шляхом подальшого підвищення привілеїв. Однак ця загроза пом'якшується рішенням мікровізора. Оскільки криптографічні ключі та сервер керування ключами розміщені у власному розділі seL4, доки фундаментальний IPC та керування пам'яттю в seL4 не зламано, навіть зловмисники з кореневими привілеями Linux не можуть отримати ключі.

У звичайних системах, якщо їх не налаштовано належним чином, зловмисний процес може підробити локальний IPC і зловживати сервером керування ключами без отримання ключів. Цей вид атаки, хоча і можливий, простіше пом'якшити в розчині мікровізора. У рішенні мікровізора, якщо шкідлива програма хоче запитати послуги від сервера керування ключами, їй потрібно буде викликати налаштований системний виклик. Модуль ядра, який обробляє системний виклик, діє як проксі-сервер і може ретельно перевіряти запит. Оскільки модуль працює в просторі ядра з високими привілеями, розробники можуть реалізувати модуль для перевірки ідентичності процесу, що викликає. До тих пір, поки ядро не скомпрометовано, а модуль ядра вставляє належний захист перевірити треба, перш ніж авторизувати будь-який запит, бо зловмисники не можуть зловживати послугами. Якщо зловмисникам вдасться скомпрометувати ядро Linux, то вони можуть підробити запит і зловживати службами так, ніби вони є власниками ключа. Однак, якщо припустити, що зловмисники можуть скомпрометувати ядро Linux, вони, по суті, заволодіють системою і можуть робити плановані дії. Таким чином, у порівнянні з цим, рішення мікровізора все ще є кращим з точки зору безпеки системи.

З іншого боку, якщо зловмисники мають фізичний доступ до пристрою, потрібно враховувати більше векторів атаки. У звичайній системі зловмисники можуть замінити завантажувальний образ і запустити шкідливу версію, за допомогою якої вони матимуть повний контроль над доступом до сховища та отриманням ключів. За підтримки безпечного завантаження такі зловмисники можуть бути усунені, оскільки непідписані образи ядра та програми не можуть бути завантажені. Для того, щоб повністю захистити систему від фізичного втручання, в обох цих рішеннях повинні застосовуватися додаткові механізми безпеки, такі як шифрування всього диска, апаратні криптографічні чіпи, наприклад, модуль довірчої платформи. Однак, такі механізми простіше застосувати в рідному розділі seL4 через модульну конструкцію в seL4, а також через те, що в такій системі немає застарілої залежності.

Таким чином, в середньому деградація віртуальної машини може становити лише 4%. Додаткові витрати на віртуалізацію значною мірою залежать від робочого навантаження та апаратної платформи. На практиці існує кілька обмежень такого підходу. По-перше, обов'язковий контроль доступу за допомогою матриці контролю доступу вимагає модифікації ядра кожного разу, коли вводяться нові політики. Частково це пов'язано з конструкцією примітивів IPC ядра MINIX 3 та структурою мікроядра. Наприклад, одним із перспективних рішень є автоматична генерація матриці контролю доступу, проксі-процесу та архітектури коду програми з моделі системи. Модель seL4, заснована на можливостях, демонструє ще один перспективний підхід. Подавши заявку на доступ на основі можливостей у ядрі, політику можна впровадити у користувачів, яка є більш гнучкою. Потрібно більше роботи з портування існуючих драйверів та впровадження бібліотек, щоб зробити систему простішою у використанні, а також оптимізувати продуктивність та ефективність. Дослідження ефективності IPC для Linux і MINIX 3 подано в табл. 1.

Ця оцінка моделює робоче навантаження системи автоматизації будівлі. Для порівняння, продуктивність мережі Linux вимірюється як базова. Результати подані є середнім арифметичним для збільшених тестів. MINIX IPC працює швидше, ніж Linux IPC при передачі невеликих фрагментів даних. MINX 3 IPC обмежений максимум 64 байтами. Однак більші передачі найбільш оптимальні через спільну пам'ять. Для мережевого зв'язку за допомогою UDP MINIX 3 працює повільніше. Це пов'язано з відсутністю загальної оптимізації продуктивності в MINIX 3 і неефективним мережевим драйвером. Однак така продуктивність все ще прийнятна в домені САБ. Для затримки мережі проксі-сервер додає лише близько 4% додаткових витрат без тунелювання та приблизно від 10% до 30% додаткових витрат при тунелюванні подібно до Linux.

Таблиця 1

IPC для Linux і MINIX 3

Розмір (байти)	Локальний IPC (мікросекунди)		
	Черга повідомлень Linux	Сокет Linux Unix	MINIX 3 IPC
1	20.5	59.6	14
64	23.4	59.3	14
128	25.2	60.4	-
256	27.3	60.7	-
512	25.1	63.2	-
1024	26.6	62.5	-

Ця оцінка не є еталоном для систематичного порівняння продуктивності ядра різних ОС. Однак отриманий результат дає попередню оцінку підходу для САБ. Запропонована структура зв'язку показує результати з додатковими витратами на продуктивність для загальних переваг безпеки зв'язку. Ґрунтуючись на цих вимірюваннях і зібраному реальному наборі даних САБ, рішення є достатнім для вимог до продуктивності САБ.

Таким чином, розроблено безпечну обчислювальну платформу для системи аутентифікації будівель наступного покоління. У процесі вивчення проблемної області проведено емпіричне дослідження двох реально існуючих систем автоматизації будівель. Проаналізовано різні потенційні ризики для безпеки і запропоновано безпечну обчислювальну платформу на основі мікроядра в спробі вирішити ці проблеми безпеки для САБ наступного покоління. З огляду на сумісність з існуючими системами, це рішення використовує архітектуру операційної системи мікроядра і застосовує обов'язковий контроль доступу та мережевий зв'язок на основі проксі-сервера для забезпечення дотримання глобальних політик в контролерах автоматизації Це рішення забезпечує зворотню сумісність.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Запропоноване рішення є надійним завдяки мінімальній функціональності мікроядра в просторі ядра та модульній архітектурі. При цьому має зворотну сумісність, оскільки застарілі програми можуть бути розміщені у віртуальному середовищі, а архітектуру проксі-сервера захищає протокол мережевого зв'язку САБ за допомогою мережевого тунелювання. Для оцінки запропонованого рішення були реалізовані різні сценарії автоматизації будівель, такі як сценарій лабораторного контролю температури, сценарій управління ключами та відповідні атаки. Ці експерименти показують переваги безпеки запропонованого рішення та відчутну ефективність.

Представлений підхід віртуалізації використовує формально перевірений seL4 як мікровізор для розміщення застарілих систем САБ. Цей підхід ґрунтується на полегшенні адаптації, одночасно збалансовуючи компроміси між витратами та безпекою. Крім того, він використовує формально перевірене мікроядро seL4 для його унікальних переваг доведеної гарантії безпеки та прогресу в технології віртуалізації. Система може розвиватися з часом, оскільки програмні продукти змінюються. Розроблено безпечну обчислювальну платформу для системи аутентифікації будівель наступного покоління за технологією кіберфізичних систем.

Напрямами подальших досліджень є удосконалення в підході, що базується на модифікаціях в програмному забезпеченні засобів IoT. Ці удосконалення стосуватимуться врахування протоколів взаємодії засобів та інтерфейсів операційних систем.

Література

1. Karnouskos S. Stuxnet Worm Impact on Industrial Cyber-Physical System Security. In *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2011.

2. Gu R., Shao Z., Chen H., Wu X., Kim J., Sjöberg V., Costanzo D. Certikos: An extensible architecture for building certified concurrent os kernels. In *OSDI*, 2016. Volume 16, P. 653–669,

3. Jaeger T. Operating system security. *Synthesis Lectures on Information Security, Privacy and Trust*, 2008.

4. Markowsky G., Savenko O., Lysenko S., Nicheporuk A. The technique for metamorphic viruses' detection based on its obfuscation features analysis. *CEUR-WS* 2104. 2018. P. 680-687.

5. Denysiuk D., Savenko O., Lysenko S., Savenko B., Kashtalian A. Method for Detecting Steganographic Changes in Images Using Machine Learning. In: *2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Athens, Greece. 2023. P. 1-6.
doi:10.1109/DESSERT61349.2023.10416453.

6. Lackorzynski A., Warg A. Timing aware hardware virtualization on the l4re microkernel system. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016. P. 67.

7. Savenko B., Lysenko S., Bobrovnikova K., Savenko O., Markowsky G.. Detection DNS Tunneling

Botnets // *Proceedings of the 2021 IEEE 11th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), IDAACS'2021*, Cracow, Poland, September 22-25, 2021.

8. Walter E., Ferguson-Walter K., Ridley A. Incorporating deception into cyberbattlesim for autonomous defense. 2021. *arXiv preprint arXiv:2108.13980*. DOI: 10.48550/arXiv.2108.13980

9. Lysenko S., Bobrovnikova K., Shchuka R., Savenko O. A Cyberattacks Detection Technique Based on Evolutionary Algorithms. *11th International Conference on Dependable Systems. Services and Technologies (DESSERT)*. 2020. V. 1. P. 127-132. DOI: 10.1109/DESSERT50317.2020.9125016

10. Rios B. Owning a Building: Exploiting Access Control and Facility Management Systems. *Black Hat Asia 2014, Singapore*, Mar 2014.

11. Klein G., Tuch H. Towards verified virtual memory in l4. *TPHOLs Emerging Trends*, 4:16, 2004.

12. Savenko B., Kashtalian A. Method for Determining the Efficiency of a Distributed Anomaly Detection System. *CSIT*. 2022. V. 2. P. 14-22. https://doi.org/10.31891/csit-2022-2-2.

13. Wang X., Mizuno M., Neilsen M., Ou X., Rajagopalan S. R., Boldwin W. G., Phillips B. Secure RTOS Architecture for Building Automation. *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy - CPS-SPC15*, Oct 2015.

14. Lysenko S., Savenko O., Bobrovnikova K., Kryshchuk A. Self-adaptive system for the corporate area network resilience in the presence of botnet cyberattacks. *Communications in Computer and Information Science.* 2018. V. 860. P. 385-401. DOI: 10.1007/978-3-319-92459-5_31.

15. Kedrowitsch A., Danfeng Y., Gang W., Cameron K. A First Look: Using Linux Containers for Deceptive Honeypots. *Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense (SafeConfig '17)*. Association for Computing Machinery, New York, NY, USA, 2017, pp. 15–22. DOI: 10.1145/3140368.3140371

16. Razali M. F., Razali M. N., Mansor F. Z., Muruti G., Jamil N. IoT Honeypot: A Review from Researcher's Perspective. *IEEE Conference on Application. Information and Network Security (AINS).* Langkawi. Malaysia, 2018. pp. 93-98. DOI: 10.1109/AINS.2018.8631494

17. Pomorova O., Savenko O., Lysenko S., Kryshchuk A., Bobrovnikova K. A Technique for the Botnet Detection Based on DNS-Traffic Analysis. In: Gaj, P., Kwiecień, A., Stera, P. (eds) *Computer Networks. CN 2015. Communications in Computer and Information Science.* 2015. V. 522. P. 127-138. DOI: 10.1007/978-3-319-19419-6_12.

18. Bobrovnikova K., Lysenko S., Savenko B., Gaj P., Savenko O. Technique for IoT malware detection based on control flow graph analysis. *Radioelectronic and Computer Systems.* 2022. V. 1. P. 141–153. DOI: 10.32620/reks.2022.1.11.

19. Lysenko S., Savenko O., Bobrovnikova K., Kryshchuk A., Savenko B. Information technology for botnets detection based on their behaviour in the corporate area network. *Communications in Computer and Information Science.* 2017. V. 718. P. 166–181. DOI: 10.1007/978-3-319-59767-6_14.

20. U.S. Department of Health, Centers for Disease Control Human Services, Public Health Service, and National Institutes of Health Prevention. *Biosafety in Microbiological and Biomedical Laboratories, 5th Edition*. HHS, 2009.

21. Lysenko S., Savenko O., Bobrovnikova, K. DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering. *CEUR-WS.* 2018. V. 2104. P. 688-695.

22. Shabtai A., Fledel Y., Elovici Y. Securing android-powered mobile devices using selinux. *IEEE Security & Privacy*, 8(3), 2010. P. 36–44.

23. Lysenko S., Savenko O., Bobrovnikova K. DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering. *CEUR-WS* 2104. 2018. P. 688-695.

24. Mathews L. Hackers Use DDoS Attack To Cut Heat To Apartments, Nov 2016.

25. Мельник В.М., Сорочинський О.Ю., Глухенький О.А., Семенюк Б.В. Метод створення безпечної кіберфізичної системи для автоматизації приміщень підприємств з використанням операційних систем на основі мікроядра / Збірник наукових праць за матеріалами XV Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2023». Хмельницький, 2023, С.190-192. https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2023-corpuspaper.pdf

References

1. Karnouskos S. Stuxnet Worm Impact on Industrial Cyber-Physical System Security. In *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2011.

2. Gu R., Shao Z., Chen H., Wu X., Kim J., Sjöberg V., Costanzo D. Certikos: An extensible architecture for building certified concurrent os kernels. In *OSDI*, 2016. Volume 16, P. 653–669,

3. Jaeger T. Operating system security. *Synthesis Lectures on Information Security, Privacy and Trust*, 2008.

4. Markowsky G., Savenko O., Lysenko S., Nicheporuk A. The technique for metamorphic viruses' detection based on its obfuscation features analysis. *CEUR-WS* 2104. 2018. P. 680-687.

5. Denysiuk D., Savenko O., Lysenko S., Savenko B., Kashtalian A. Method for Detecting Steganographic Changes in Images Using

Machine Learning. In: *2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Athens, Greece. 2023. P. 1-6.
doi:10.1109/DESSERT61349.2023.10416453.

6. Lackorzynski A., Warg A. Timing aware hardware virtualization on the l4re microkernel system. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016. P. 67.

7. Savenko B., Lysenko S., Bobrovnikova K., Savenko O., Markowsky G.. Detection DNS Tunneling Botnets // *Proceedings of the 2021 IEEE 11th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), IDAACS'2021*, Cracow, Poland, September 22-25, 2021.

8. Walter E., Ferguson-Walter K., Ridley A. Incorporating deception into cyberbattlesim for autonomous defense. 2021. *arXiv preprint arXiv:2108.13980*. DOI: 10.48550/arXiv.2108.13980

9. Lysenko S., Bobrovnikova K., Shchuka R., Savenko O. A Cyberattacks Detection Technique Based on Evolutionary Algorithms. *11th International Conference on Dependable Systems. Services and Technologies (DESSERT).* 2020. V. 1. P. 127-132. DOI: 10.1109/DESSERT50317.2020.9125016

10. Rios B. Owning a Building: Exploiting Access Control and Facility Management Systems. *Black Hat Asia 2014, Singapore*, Mar 2014.

11. Klein G., Tuch H. Towards verified virtual memory in l4. *TPHOLs Emerging Trends*, 4:16, 2004.

12. Savenko B., Kashtalian A. Method for Determining the Efficiency of a Distributed Anomaly Detection System. *CSIT*. 2022. V. 2. P. 14-22. https://doi.org/10.31891/csit-2022-2-2.

13. Wang X., Mizuno M., Neilsen M., Ou X., Rajagopalan S. R., Boldwin W. G., Phillips B. Secure RTOS Architecture for Building Automation. *Proceedingsof the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy - CPS-SPC15*, Oct 2015.

14. Lysenko S., Savenko O., Bobrovnikova K., Kryshchuk A. Self-adaptive system for the corporate area network resilience in the presence of botnet cyberattacks. *Communications in Computer and Information Science*. 2018. V. 860. P. 385-401. DOI: 10.1007/978-3-319-92459-5_31.

15. Kedrowitsch A., Danfeng Y., Gang W., Cameron K. A First Look: Using Linux Containers for Deceptive Honeypots. *Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense (SafeConfig '17)*. Association for Computing Machinery, New York, NY, USA, 2017, pp. 15–22. DOI: 10.1145/3140368.3140371

16. Razali M. F., Razali M. N., Mansor F. Z., Muruti G., Jamil N. IoT Honeypot: A Review from Researcher's Perspective. *IEEE Conference on Application. Information and Network Security (AINS).* Langkawi. Malaysia, 2018. pp. 93-98. DOI: 10.1109/AINS.2018.8631494

17. Pomorova O., Savenko O., Lysenko S., Kryshchuk A., Bobrovnikova K. A Technique for the Botnet Detection Based on DNS-Traffic Analysis. In: Gaj, P., Kwiecień, A., Stera, P. (eds) *Computer Networks. CN 2015. Communications in Computer and Information Science.* 2015. V. 522. P. 127-138. DOI: 10.1007/978-3-319-19419-6_12.

18. Bobrovnikova K., Lysenko S., Savenko B., Gaj P., Savenko O. Technique for IoT malware detection based on control flow graph analysis. *Radioelectronic and Computer Systems*. 2022. V. 1. P. 141–153. DOI: 10.32620/reks.2022.1.11.

19. Lysenko S., Savenko O., Bobrovnikova K., Kryshchuk A., Savenko B. Information technology for botnets detection based on their behaviour in the corporate area network. *Communications in Computer and Information Science*. 2017. V. 718. P. 166–181. DOI: 10.1007/978-3-319-59767-6_14.

20. U.S. Department of Health, Centers for Disease Control Human Services, Public Health Service, and National Institutes of Health Prevention. *Biosafety in Microbiological and Biomedical Laboratories, 5th Edition*. HHS, 2009.

21. Lysenko S., Savenko O., Bobrovnikova, K. DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering. *CEUR-WS*. 2018. V. 2104. P. 688-695.

22. Shabtai A., Fledel Y., Elovici Y. Securing android-powered mobile devices usingselinux. *IEEE Security & Privacy*, 8(3), 2010. P. 36–44.

23. Lysenko S., Savenko O., Bobrovnikova K. DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering. *CEUR-WS* 2104. 2018. P. 688-695.

24. Mathews L. Hackers Use DDoS Attack To Cut Heat To Apartments, Nov 2016.

25. Melnyk V.M., Sorochynskyi O.Yu., Gluchenkyi O.A., Semenyuk B.V. The method of creating a safe cyber-physical system for the automation of enterprise premises using microkernel-based operating systems / Collection of scientific papers based on the materials of the XV All-Ukrainian scientific and practical conference "Actual problems of computer science APKN-2023". Khmelnytskyi, 2023, P. 190-192. https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2023-corpuspaper.pdf (In Ukrainen)