УДК 004.05 **DOI: 10.20998/2079-0023.2020.01.12**

А. О. ГАПОН, В. М. ФЕДОРЧЕНКО, А. О. ПОЛЯКОВ, В. Ю. ВОЛОВЩИКОВ, В. О. ГУЖВА

АНАЛІЗ МЕТОДОЛОГІЇ DEVSECOPS В ПРОЦЕСАХ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Предметом дослідження в статті є методологія розробки і захисту програмного забезпечення в рамках DevSecOps. Дана методологія змінила підхід до забезпечення безпеки з реактивного на проактивний, а також підкреслює важливість забезпечення безпеки на всіх рівнях організації. DevSecOps означає забезпечення безпеки в розробці додатків від самих ранніх етапів до самого кінця, а також включає в себе автоматизацію деяких шлюзів безпеки, щоб запобігти уповільнення робочого процесу DevOps. Необхідно підтримувати короткі і часто повторювані цикли розробки програмного продукту, а також інтегрувати заходи безпеки. Вибір правильних інструментів для безперервної інтеграції безпеки може допомогти в досягненні цих цілей. Сучасні інструменти автоматизації допомогли організаціям впровадити більш гнучкі методи розробки, а також зіграли свою роль в розробці нових заходів безпеки. Для ефективного захисту DevOps потрібні не тільки нові інструменти, а й зміни в самій організації процесів DevOps, щоб швидше інтегрувати роботу груп фахівців з безпеки з іншими спеціалістами, що призведе до покращення якості продукту. Стаття присвячена детальному аналізу сучасних підходів і методологій систематизації розробки та захисту програмного забезпечення, серед яких SDLC, BSIMM і OpenSAMM. Мета роботи – класифікація підходів до побудови процесів DevSecOps, а також розгляд методологій систематизації існуючих засобів захисту програмного забезпечення, що забезпечують взаємодію команди розробників і фахівців із захисту інформації в рамках одного життєвого циклу розробки. У статті вирішуються наступні завдання: розгляд і аналіз підходів побудови процесів DevSecOps і розгляд методологій систематизації засобів захисту програмного забезпечення. Отримані наступні результати: проаналізовано необхідні складові для побудови DevSecOps процесів. Висновки: проведений аналіз дозволяє класифікувати процес розробки і захисту програмного забезпечення за допомогою методології DevSecOps.

Ключові слова: DevSecOps, Application security, Infrastructure security, SDLC, BSIMM, OpenSAMM.

А. А. ГАПОН, В. М. ФЕДОРЧЕНКО, А. А. ПОЛЯКОВ, В. Ю. ВОЛОВЩИКОВ, В. А. ГУЖВА

АНАЛИЗ МЕТОДОЛОГИИ DEVSECOPS В ПРОЦЕССАХ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Предметом исследования в статье является методология разработки и защиты программного обеспечения в рамках DevSecOps. Данная методология изменила подход к обеспечению безопасности с реактивного на проактивный, а также подчеркивает важность обеспечения безопасности на всех уровнях организации. DevSecOps означает обеспечение безопасности в разработке приложений от самых ранних этапов до самого конца, а также включает в себя автоматизацию некоторых шлюзов безопасности, чтобы предотвратить замедление рабочего процесса DevOps. Необходимо поддерживать короткие и часто повторяемые циклы разработки программного продукта, а также интегрировать меры безопасности. Выбор правильных инструментов для непрерывной интеграции безопасности может помочь в достижении этих целей. Современные инструменты автоматизации помогли организациям внедрить более гибкие методы разработки, а также сыграли свою роль в разработке новых мер безопасности. Для эффективной защиты DevOps требуются не только новые инструменты, но и изменения в самой организации процессов DevOps, чтобы быстрее интегрировать работу групп специалистов по безопасности с другими специалистами, что приведет к улучшению качества продукта. Статья посвящена детальному анализу современных подходов и методологий систематизации разработки и защиты программного обеспечения, среди которых SDLC, BSIMM и OpenSAMM. Цель работы – классификация подходов к построению процессов DevSecOps, а также рассмотрение методологий систематизации существующих средств защиты программного обеспечения, обеспечивающих взаимодействие команды разработчиков и специалистов по защите информации в рамках одного жизненного цикла разработки. В статье решаются следующие задачи: рассмотрение и анализ подходов построения процессов DevSecOps и рассмотрение методологий систематизации средств защиты программного обеспечения. Получены следующие результаты: проанализированы необходимые составляющие для построения DevSecOps процессов. Выводы: проведенный анализ позволяет классифицировать процесс разработки и защиты программного обеспечения с помощью методологии DevSecOps.

Ключевые слова: DevSecOps, Applications ecurity, Infrastructure security, SDLC, BSIMM, OpenSAMM.

A. O. HAPON, V. M. FEDORCHENKO, A. O. POLIAKOV, V. Y. VOLOVSHCHYKOV, V. A. GUZHVA

ANALYSIS OF DEVSECOPS METHODOLOGY IN SOFTWARE DEVELOPMENT PROCESSES

The subject of this research is the software development and protection methodology within DevSecOps. This methodology has changed the approach to ensuring security from reactive to proactive, and also emphasizes the importance of security at all levels of the organization. DevSecOps means providing security in application development from the earliest stages to the very end, and also includes automating some security gateways to prevent DevOps from slowing down the workflow. It is necessary to maintain short and frequently repeated cycles of software product development, as well as integrate security measures. Choosing the right tools for continuous security integration can help achieve these goals. Modern automation tools have helped organizations implement more flexible development methods, and also played a role in the development of new security measures. Effective protection of DevOps requires not only new tools, but also changes in the organization of DevOps processes in order to quickly integrate the work of security teams with other specialists, which will improve the quality of the product. The article is devoted to a detailed analysis of modern approaches and methodologies for systematizing software development and protection, including SDLC, BSIMM and OpenSAMM. The purpose of the work is the classification of approaches to the construction of DevSecOps processes, as well as the consideration of systematization methodologies for existing software protection tools that ensure the interaction of a development team and information protection specialists within one development life cycle. The following tasks are solved in the article: consideration and analysis of DevSecOps process construction approaches and consideration of systematization methodologies for software protection tools. The following results were obtained: the necessary components for the construction of DevSecOps processes are analyzed. Conclusions: the analysis allows us to classify the process of developing and protecting software using the DevSecOps methodology.

Keywords: DevSecOps, Application security, Infrastructure security, SDLC, BSIMM, OpenSAMM.

Вступ. DevOps описує набір практик, які автоматизують процеси між розробниками програмного забезпечення, щоб вони могли швидше і надійніше створювати, тестувати і випускати програмне забезпечення.

DevSecOps – одна з найважливіших тенденцій DevOps. Це підхід до безпеки операцій, що дозволяє використовувати принципи і кращі практики DevOps для забезпечення кращої, швидкості більш безпечної доставки програмного забезпечення. По суті, це означає, що всі вимоги безпеки з самого початку кодифіковані, а контроль безпеки і розробка здійснюються паралельно, причому безпеку намагаються впровадити в кожну частину процесу agile-розробки. Завдяки цьому DevSecOps може знизити витрати пов'язані з виправленням недоліків безпеки [1].

Цей підхід вигідно відрізняється від моделі того, що було прийнято до DevSecOps, де контроль безпеки був заключним процесом і здійснювався в кінці розробки.

На рис. 1 зображені основні складові DevOps і DevSecOps.

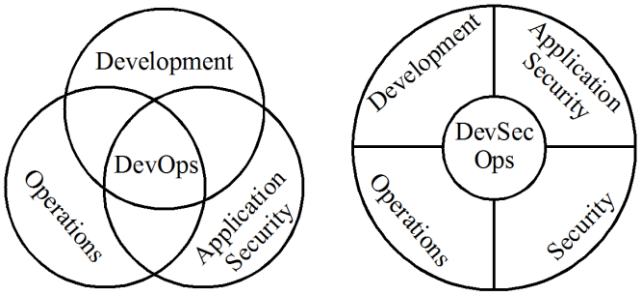

Рис. 1. Порівняння DevOps з DevSecOps

Як і в DevOps, DevSecOps не концентрує налагодження безпеки в одному місці. Весь процес залежить від усіх команд, що беруть участь в розробці. Тобто, робота за двома методологіями ведеться паралельно одними і тими ж учасниками. І тому, цілком можливо, що в найближчому майбутньому їх і зовсім не розділятимуть.

Раннє впровадження гарантування безпеки забезпечує кращу кодову базу і більш високий рівень безпеки. Більш активний підхід до виявлення помилок і дефектів знижує уразливість. Також можна реагувати на інциденти значно швидше, якщо почати розробку свого продукту з урахуванням ризиків.

Рання інтеграція інструментів безпеки в процес розробки програмного забезпечення забезпечує кращий продукт. При запуску автоматичних тестів також виконуються тести безпеки.

Зазвичай, методики для оптимізації процесів розробки програмного забезпечення націлені виключно на підвищення ефективності всередині команди, але в DevSecOps мова йде про застосування автоматизованих інструментів для гарантування комплексного захисту.

Варто зазначити, що кожна з доступних методик стрімко прискорює роботу, жертвуючи при цьому безпекою інфраструктури. Більшість компаній може бути не готова до такого стрибка підвищення вимог якості в даній сфері.

Саме тому, подальший розвиток DevOps порушив питання інформаційної безпеки. Прискорення роботи команд-розробників створило безперервний потік оновлюваних функцій, а також постійний потік даних з боку сервісів, користувачів та інших додатків [2].

Розгортання коду має відбуватися частіше і завершуватися за менший час. Більш короткий час циклу є ознакою оптимізованих процесів, в той час як більш тривалий час може бути ознакою того, що необхідно переглянути свої кращі практики або інструменти кодування.

На рис. 2 зображені етапи циклу розробки DevSecOps.

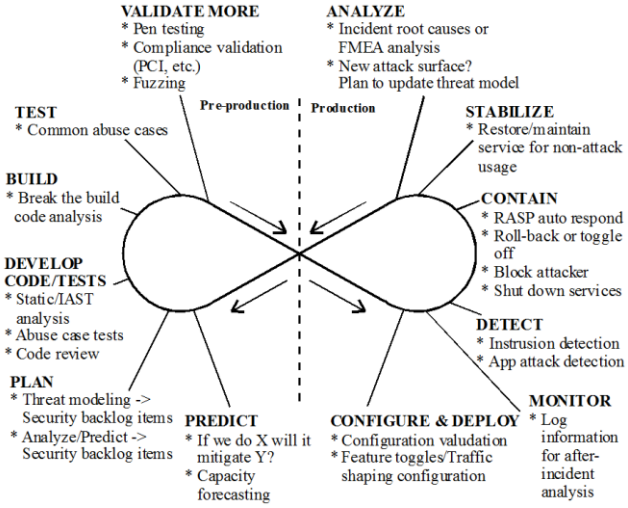

Рис. 2. Цикл розробки DevSecOps

Ризики неякісного коду включають погрози продуктивності і безпеки, які можуть бути дорогими як для компанії, так і для споживача. Код хорошої якості повинен бути послідовним, добре документованим і тестуємим.

DevSecOps може використовуватися, наприклад, при переході на мікросервіси, в процесах Безперервною інтеграції (Continuous Integration, CI) і Безперервного розгортання (Continuous Deployment, CD), або просто для тестування хмарної інфраструктури. Набор конкретних рішень залежить від використовуваного технічного стека і архітектури [3].

Підходи до побудови DevSecOps. Application Security – це розділ безпеки, який відповідає за безпеку додатку. Це не відноситься до інфраструктури або до мережевої безпеки, а саме до того, що пишуть і над чим працюють розробники – це недоліки і уразливості самого додатка [4].

На рис. 3 зображені складові Application Security.

Endpoint Security (безпека кінцевих точок) – відноситься до захисту кінцевих пристроїв кінцевого користувача, таких як настільні комп'ютери, ноутбуки і мобільні пристрої. Кінцеві точки служать точками доступу до корпоративної мережі і створюють точки входу, які можуть бути використані зловмисниками. Endpoint Security захищає ці точки входу від ризикованих дій або зловмисних атак [5].

Content Security Policy (політика захисту контенту, CSP) – це механізм забезпечення безпеки, за допомогою якого можна захищатися від атак з впровадженням контенту, наприклад, міжсайтового скриптинга (XSS, cross site scripting). CSP описує

безпечні джерела завантаження ресурсів, встановлює правила використання вбудованих стилів, скриптів, а також завантаження з ресурсів, що не входять до «білого списку», блокуються [6].

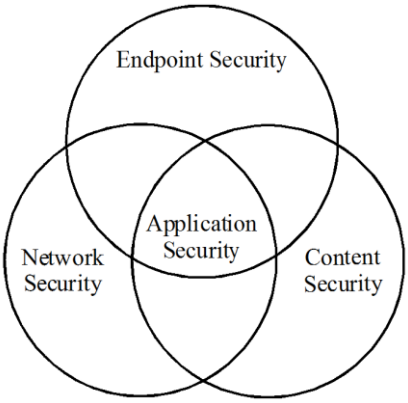

Рис. 3. Складові Application Security

Також важливою складовою DevSecOps є Infrastructure Security. Ключовою сферою знань, яка життєво важлива для будь-якого спеціаліста з безпеки, є чітке розуміння IT-інфраструктури та її взаємозв'язку зі створенням комплексної стратегії безпеки.

Infrastructure Security – це заходи безпеки для захисту інфраструктури, особливо критично важливої інфраструктури, такої як мережеві комунікації, центр зв'язку, серверний центр, центр баз даних і IT-центр. Безпека інфраструктури прагне обмежити вразливість цих структур і систем.

Network Security – положення і політики, прийняті адміністратором мережі для запобігання та моніторингу несанкціонованого доступу, неправильного використання, модифікації або відмови в комп'ютерній мережі. Це передбачає авторизацію доступу до даних в мережі, яка контролюється адміністратором.

Cryptography-Digital Forensics – процес кодування повідомлень або інформації таким чином, що для підслуховуючі або хакери не можуть прочитати і досліджувати цифрові носії криміналістично з метою виявлення, збереження, відновлення, аналізу та подання фактів і думки про інформацію.

Information Security – політика і стратегія захисту інформації від несанкціонованого доступу, використання, розкриття, порушення, модифікації, прочитання, перевірки, записи або знищення. Уряд, військові, корпорації, фінансові установи, лікарні і приватні підприємства накопичують багато конфіденційної інформації про своїх співробітників, клієнтів, продуктах, дослідженнях і фінансовий стан [7].

Команда мережевої безпеки реалізує апаратне і програмне забезпечення, необхідне для захисту архітектури безпеки. При наявності належної мережевої безпеки система може виявляти виникаючі загрози, перш ніж вони проникнуть в мережу і скомпрометують дані. Існує безліч компонентів системи безпеки мережі, які працюють разом, щоб поліпшити стан безпеки [8].

Існує багато кроків, які необхідно вжити, щоб запобігти або хоча б зменшити кількість і серйозність вразливостей в додатках. Звичайно, найкраще виявляти уразливості на ранніх етапах життєвого циклу розробки програмного забезпечення («зсув вліво»), тому що тоді вартість буде меншою. Аналіз і огляд коду грають тут ключову роль. Однак, оскільки не всі проблеми можуть бути виявлені на етапі розробки додатку, зазвичай під час запуску програми через різні шлюзи тестування безпеки додатки перед його випуском і, можливо, обертається і захищається по різному. Після релізу можна додатково захистити додаток, перевіряючи його вхідні дані.

Security development lifecycle. Цикл розробки пропонує шаблон, використування якого полегшує проектування, створення і випуск якісного програмного забезпечення. Це концепція, що визначає процеси і засоби, необхідні для успішного завершення проекту. Метою використання моделі життєвого циклу є створення ефективного, економічно вигідного і якісного програмного продукту.

Security development lifecycle (SDL або SDLC) – концепція розробки, яка полягає у формуванні вимог до додатка, безпечному програмуванні, тестуванні, сертифікації, експлуатації та оновлення. SDL був розроблений компанією Microsoft [9].

На рис. 4 – канонічна модель SDLC, основне завдання якої визначити участь безпеки на кожному етапі розробки, від вимог, до релізу і виходу в експлуатацію.

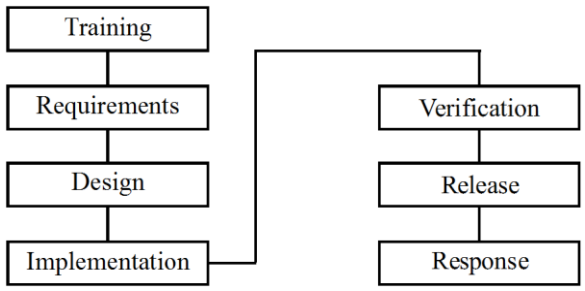

Рис. 4. Security development lifecycle (SDL)

Досвід компанії Microsoft показує, що SDL ефективний у зниженні числа вразливостей безпеки. Первісна реалізація і поступове впровадження елементів складових SDL призвело до значного поліпшення безпеки програмного забезпечення.

Розробка і впровадження життєвого циклу розробки безпеки являє собою серйозну інвестицію для Microsoft і серйозне змінення в тому, як програмне забезпечення проектується, розробляється і тестується [10].

Application Security і SDLC спрямовані не на виявлення вразливостей, а на запобігання їх появи. Згодом канонічний SDLC від Microsoft був сильно деталізований в різних методологіях – OpenSAMM, BSIMM.

Building Security In Maturity Model. BSIMM описує методологію систематизації існуючих заходів забезпечення безпеки в галузі програмного забезпечення. Кількісно оцінюючи практику багатьох різних

організацій, за допомогою BSIMM можна описати підходи що поділяються багатьма, а також відмінності, які роблять кожен з них унікальним.

BSIMM дозволяє побудувати довгостроковий план по гарантуванню безпеки програмного забезпечення і відстеження просування відповідно до цього плану.

Основа методології – поділ процесу Application Security на 4 домена: Governance, Intelligence, SSDL Touchpoints і Deployment. У кожному домені 12 практик, які представлені у вигляді 119 активностей.

На рис. 5 зображені складові методології систематизації підходів гарантування безпеки програмного забезпечення BSIMM.

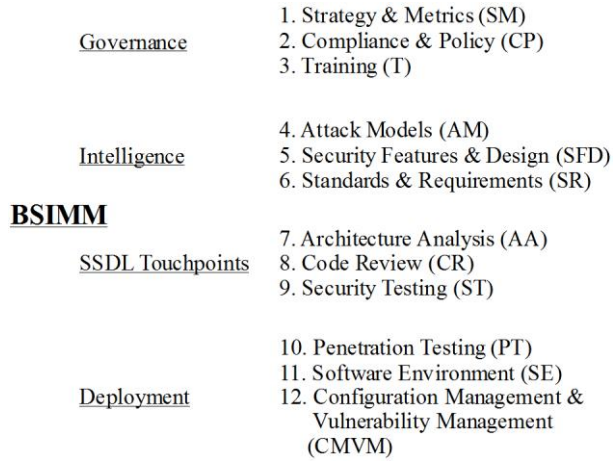

Рис. 5. Building security in maturity model (BSIMM)

Governance – це практики, які допомагають організовувати, управляти і вимірювати ініціативу щодо забезпечення безпеки програмного забезпечення.

Intelligence – це практики, які призводять до накопичення корпоративних знань, використовуваних при проведенні заходів щодо забезпечення безпеки програмного забезпечення в усій організації.

SSDL Touchpoint – це практики, пов'язані з аналізом і перевіркою конкретних артефактів і процесів розробки програмного забезпечення.

Deployment – це практики, які взаємодіють з традиційною мережевою безпекою та підтримки програмного забезпечення [11].

У кожної з 119 активностей є 3 рівня зрілості: початковий, середній і просунутий. Усі 12 практик можна вивчати по розділах, відбирати важливі елементи, розбиратися, як їх впроваджувати і поступово додавати елементи, наприклад, статичний і динамічний аналіз коду або огляд коду [12].

OpenSAMM. Модель зрілості Software Assurance (SAMM) описує методологію систематизації, котра допомагає організаціям формулювати і реалізовувати стратегію забезпечення безпеки програмного забезпечення, адаптовану до конкретних ризиків, з якими стикається організація [13].

В основі моделі лежать основні бізнес-функції розробки програмного забезпечення, до яких прив'язані методи забезпечення безпеки

Будівельні блоки моделі – це три рівня зрілості, певні для кожної з дванадцяти практик забезпечення безпеки. Вони визначають широкий спектр можливостей, який організація може задіяти, щоб знизити ризики безпеки і підвищити надійність програмного забезпечення.

На рис. 6 зображені складові методології систематизації підходів гарантування безпеки програмного забезпечення SAMM.

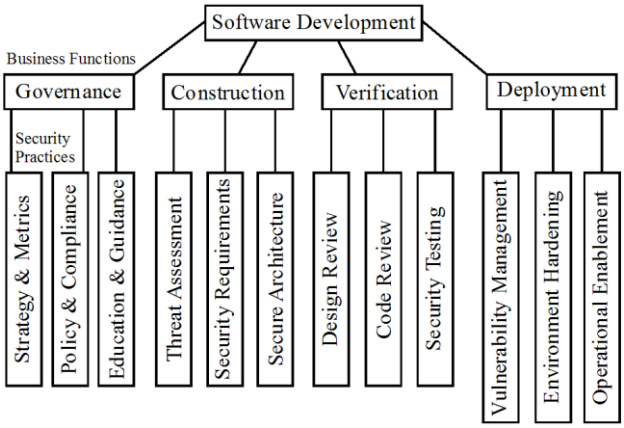

Рис. 6. Software Assurance Maturity Model (SAMM)

На найвищому рівні SAMM визначає чотири критичних бізнес-функції. Кожна бізнес-функція являє собою категорію дій, пов'язаних з основними моментами розробки програмного забезпечення [14].

Для кожної бізнес-функції SAMM визначає три практики безпеки. Кожна практика безпеки є областю діяльності, що пов'язана з безпекою, яка створює гарантії для відповідної бізнес-функції.

Таким чином, в цілому існує дванадцять практик безпеки, які є незалежними сховищами для поліпшень, які відображаються в бізнес-функції розробки програмного забезпечення.

Governance – це те, як організація управляє діяльністю з розробки програмного забезпечення. Він забезпечує відносини між ефективними бізнес-процесами і групами розробників, забезпечуючи ефективний вимірний процес і правила взаємодії.

Construction – це те, що стосується процесів і діяльності, пов'язаних з бізнес-цілями і проектами розвитку. В цілому, він визначає процес створення програми, який включає дії, пов'язані з управлінням продуктом, збором вимог, специфікацією архітектури високого рівня, детальним проектуванням і аналізом реалізації.

Verification – це перевірка і тестування програмного забезпечення. Основна увага приділяється процесам і діям, пов'язаним з тим, як організація аналізує і тестує створені об'єкти в ході розробки програмного забезпечення.

Deployment – це бізнес-вимоги до розгортання, що описують процеси і дії, пов'язані з процедурами випуску програмного забезпечення. Сюди також входить спосіб доставки продуктів кінцевим користувачам, розгортання таких саме і очікуваних операцій в середовищі виконання.

Для кожної практики безпеки SAMM визначає три рівні зрілості як цілі. Кожен рівень в практиці безпеки характеризує послідовно більш складні цілі, що визначають конкретні дії і більш суворі метрики, ніж на попередньому рівні. Крім того, кожна практика безпеки може бути поліпшена незалежно, хоча пов'язані дії можуть привести до оптимізації.

Висновки. DevSecOps дозволяє організації застосовувати попереджуючий підхід до безпеки. Це спонукає розробників програмного забезпечення інтегрувати безпеку в свої повсякденні зусилля. У той же час групи безпеки можуть працювати з розробниками програмного забезпечення, щоб допомогти організації виявити і усунути вразливості безпеки, перш ніж вони вийдуть з-під контролю.

DevSecOps змінює безпеку з реактивної на проактивну, а також підкреслює важливість безпеки на всіх рівнях організації, і уповноважує співробітників служби безпеки приймати рішення, які мають позитивний вплив на їхній бізнес. Таким чином, DevSecOps, як концепція і практика, весь час розвивається, зі збільшенням кількості організацій, які впроваджують DevSecOps як рішення для їх проблеми безпеки [15].

Попит на DevSecOps збільшиться в організаціях всіх розмірів і у всіх галузях. У міру того, як все більше організацій шукають способи виявлення та виправлення проблем безпеки на ранніх етапах процесу розробки програмного забезпечення, попит на інструменти для підтримки DevSecOps відповідно збільшуватиметься.

Організація, яка впроваджує інструменти DevSecOps сьогодні, може пожинати плоди цих інвестицій на роки вперед. Надаючи розробникам програмного забезпечення і командам безпеки зручні та ефективні інструменти DevSecOps, організація розвиває культуру співпраці, спілкування, прозорості та відкритості. В результаті ця організація створює середовище, в якій розробники та групи безпеки постійно удосконалюються.

Переваги, які DevSecOps приносить компаніям це – зниження витрат, збільшення швидкості доставки, швидкості відновлення, відповідність в масштабі і пошуку загроз. Сукупний ефект цих переваг – це підвищення ділової репутації та більш плавна бізнес-модель. DevSecOps успішно видалить бар'єри між DevOps і Security, яка заважають їм працювати як єдине ціле.

DevSecOps матиме можливість знаходити і виправляти проблеми безпеки на початку процесу розробки, тим самим значно скорочуючи витрати, пов'язані з їх виявленням і виправленням.

Дуже важливо включити гарантування безпеки в життєвий цикл розробки Agile. Завдяки DevSecOps розробники можуть краще зрозуміти критичність уразливостей, які існують у їхньому коді, і виправити ці вразливості, надаючи швидкі, але більш безпечні продукти або рішення. Оскільки підхід DevSecOps автоматизований, тому команді розробників більше не потрібно записувати правила безпеки у свій код. DevSecOps знижує ризик перенапруження даних, оптимально застосовуючи ресурси.

Список літератури

1. *ITFB*. URL: https://itfb.com.ua/chto-takoe-devsecops (дата звернення: 2.11.2019).
2. *System-Admins*. URL: https://system-admins.ru/razrabotka-zashhishhennyx-prilozhenij-s-pomoshhyu-devsecops (дата звернення: 4.11.2019).
3. *Antimalware*. URL: https://www.anti-malware.ru/analytics/ Technology_Analysis/what-is-devsecops-developing-more-secure-applications#part5 (дата звернення: 2.11.2019).
4. *Newcontext*. URL: https://www.newcontext.com/what-is-devsecops (дата звернення: 2.11.2019).
5. *Forcepoint*. URL: https://www.forcepoint.com/cyber-edu/endpoint-security (дата звернення: 10.11.2019).
6. *Habr*. URL: https://habr.com/ru/company/nix/blog/271575 (дата звернення: 12.11.2019).
7. *IGI-Global*. URL: https://www.igi-global.com/chapter/managing-compliance-with-an-information-security-management-standard/112547 (дата звернення: 15.11.2019).
8. *Secureworks*. URL: https://www.secureworks.com/blog/cybersecurity-vs-network-security-vs-information-security (дата звернення: 4.11.2019).
9. *Microsoft*. URL: https://docs.microsoft.com/en-us/previous-versions/ms995349(v=msdn.10)?redirectedfrom=MSDN (дата звернення: 20.11.2019).
10. *BSIMM*. URL: https://www.bsimm.com/about.html (дата звернення: 23.11.2019).
11. *Synopsys*. URL: https://www.synopsys.com/software-integrity/resources/knowledge-database/what-is-bsimm.html (дата звернення: (20.11.2019).
12. *Habr*. URL: https://habr.com/ru/company/oleg-bunin/blog/448488 (дата звернення: 20.11.2019).
13. *OpenSAMM*. URL: https://opensamm.org/downloads/SAMM-1.0-en_US.pdf (дата звернення: 20.11.2019).
14. *Hack2Secure*. URL: https://www.hack2secure.com/blogs/summarizing-open-software-assurance-maturity-model-opensamm-requirements (дата звернення: 21.11.2019).
15. *DevSecOps Whitepaper*. URL: https://www.devseccon.com/wp-content/uploads/2017/07/DevSecOps-whitepaper.pdf (дата звернення: 23.11.2019).

References (transliterated)

1. *ITFB*. URL: https://itfb.com.ua/chto-takoe-devsecops (access date: 2.11.2019).
2. *System-Admins*. URL: https://system-admins.ru/razrabotka-zashhishhennyx-prilozhenij-s-pomoshhyu-devsecops (access date: 4.11.2019).
3. *Antimalware*. URL: https://www.anti-malware.ru/analytics/Technology_Analysis/what-is-devsecops-developing-more-secure-applications#part5 (access date: 2.11.2019).
4. *Newcontext*. URL: https://www.newcontext.com/what-is-devsecops (access date: 2.11.2019).
5. *Forcepoint*. URL: https://www.forcepoint.com/cyber-edu/endpoint-security (access date: 10.11.2019).
6. *Habr*. URL: https://habr.com/ru/company/nix/blog/271575 (access date: 12.11.2019).
7. *IGI-Global*. URL: https://www.igi-global.com/chapter/managing-compliance-with-an-information-security-management-standard/112547 (access date: 15.11.2019).
8. *Secureworks*. URL: https://www.secureworks.com/blog/cybersecurity-vs-network-security-vs-information-security (access date: 4.11.2019).
9. *Microsoft*. URL: https://docs.microsoft.com/en-us/previous-versions/ms995349(v=msdn.10)?redirectedfrom=MSDN (access date: 20.11.2019).
10. *BSIMM*. URL: https://www.bsimm.com/about.html (access date: 23.11.2019).
11. *Synopsys*. URL: https://www.synopsys.com/software-integrity/resources/knowledge-database/what-is-bsimm.html (access date: (20.11.2019).
12. *Habr*. URL: https://habr.com/ru/company/oleg-bunin/blog/448488 (access date: 20.11.2019).

13. *OpenSAMM*. URL: https://opensamm.org/downloads/SAMM-1.0-en_US.pdf (access date: 20.11.2019).
14. *Hack2Secure*. URL: https://www.hack2secure.com/blogs/summarizing-open-software-assurance-maturity-model-opensamm-requirements (access date: 21.11.2019).
15. *DevSecOps Whitepaper*. URL: https://www.devseccon.com/wp-content/uploads/2017/07/DevSecOps-whitepaper.pdf (access date: 23.11.2019).

Надійшла (received) 02.04.2020

Відомості про авторів / Сведения об авторах / About the Authors

Гапон Андрій Олександрович – Харківський національний університет радіоелектроніки, аспірант кафедри безпеки інформаційних технологій; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-2560-7426; e-mail: gapon.andrei@gmail.com

Федорченко Володимир Миколайович – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри безпеки інформаційних технологій; м. Харків, Україна; ORCID: http://orcid.org/0000-0001-7359-1460; e-mail: fedorchenko.fedor@gmail.com

Поляков Андрій Олександрович – кандидат технічних наук, доцент, Харківський національний економічний університет імені Семена Кузнеця, доцент кафедри інформаційних систем; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-1805-9011; e-mail: polyakov.andrey@gmail.com

Воловщиков Валерій Юрійович – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри програмної інженерії та інформаційних технологій управління; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-4454-2314; e-mail: valera@kpi.kharkov.ua

Гужва Віктор Олексійович – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», професор кафедри програмної інженерії та інформаційних технологій управління; м. Харків, Україна; ORCID: https://orcid.org/0000-0001-6832-4480; e-mail: guzhva.v.a@gmail.com

Гапон Андрей Александрович – Харьковский национальный университет радиоэлектроники, аспирант кафедры безопасности информационных технологий; г. Харьков, Украина; ORCID: https://orcid.org/0000-0003-2560-7426; e-mail: gapon.andrei@gmail.com

Федорченко Владимир Николаевич – кандидат технических наук, доцент, Харьковский национальный университет радиоэлектроники, доцент кафедры безопасности информационных технологий; г. Харьков, Украина; ORCID: http://orcid.org/0000-0001-7359-1460; e-mail: fedorchenko.fedor@gmail.com

Поляков Андрей Александрович – кандидат технических наук, доцент, Харьковский национальный экономический університет, доцент кафедры информационных систем; г. Харьков, Украина; ORCID: https://orcid.org/0000-0003-1805-9011; e-mail: polyakov.andrey@gmail.com

Воловщиков Валерий Юрьевич – кандидат технических наук, доцент, Национальный технический университет «Харьковский политехнический институт», доцент кафедры программной инженерии и информационных технологий управления; г. Харьков, Украина; ORCID: https://orcid.org/0000-0003-4454-2314; e-mail: valera@kpi.kharkov.ua

Гужва Виктор Алексеевич – кандидат технических наук, доцент, Национальный технический университет «Харьковский политехнический институт», профессор кафедры программной инженерии и информационных технологий управления; г. Харьков, Украина; ORCID: https://orcid.org/0000-0001-6832-4480; e-mail: guzhva.v.a@gmail.com

Hapon Andrii Oleksandrovich – Kharkiv National University of Radio Electronics, Postgraduate of Department of Information Technology Security; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0003-2560-7426; e-mail: gapon.andrei@gmail.com

Fedorchenko Volodymyr Mykolayovych – Candidate of Engineering Sciences, Docent, Kharkiv National University of Radio Electronics, Associate Professor of the Department of Information Technology Security; Kharkiv, Ukraine; ORCID: http://orcid.org/0000-0001-7359-1460; e-mail: fedorchenko.fedor@gmail.com

Polyakov Andrii Oleksandrovich – Candidate of Engineering Sciences, Docent, Semen Kuznets Kharkiv National Economic University, Associate Professor of the Department of Information Systems; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0003-1805-9011; e-mail: polyakov.andrey@gmail.com

Volovshchykov Valeriy Yuriyovich – Candidate of Technical Sciences, Docent, National Technical University "Kharkiv Polytechnic Institute", Associate Professor of the Department of Software Engineering and Management Information Technologies; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0003-4454-2314; e-mail: valera@kpi.kharkov.ua

Guzhva Viktor Alexeevich – Candidate of Technical Sciences, Docent, National Technical University "Kharkiv Polytechnic Institute", Professor of the Department of Software Engineering and Management Information Technologies; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0001-6832-4480; e-mail: guzhva.v.a@gmail.com