DOI: 10.58254/viti.5.2024.18.204

УДК 681.35

Терещенко Т. П. ORCID: 0000-0002-9659-7897 (ВІТІ ім. Героїв Крут)
канд. техн. наук Остапчук В. М. ORCID: 0000-0001-5686-0198 (ВІТІ ім. Героїв Крут)
канд. техн. наук Хусаінов П. В. ORCID: 0000-0002-0675-0369 (ВІТІ ім. Героїв Крут)
Черниш Ю. О. ORCID: 0000-0002-6626-5656 (ВІТІ ім. Героїв Крут)

ОЦІНКА ТА ЗАПОБІГАННЯ ПРОЯВУ ВРАЗЛИВОСТІ SERVER-SIDE WEB APPLICATION

Вразливість (vulnerability) системи (технологічної, комунікаційної) об'єкта кіберзахисту до здійснення негативного технічного ефекту (negative technical impact) завжди базується на експлуатації її дефектів. Дефект (weakness) – властивість програмних, апаратних, програмно-апаратних або системних компонентів, які за певних умов можуть призвести до вразливості. Атака (attack) – спроба експлуатації дефекту (дефектів) системи (об'єкта кіберзахисту) для здійснення негативного технічного ефекту (несанкціоноване читання/запис даних, неправильна робота, обхід захисних механізмів, аномальне споживання ресурсів; виконання нав'язаних команд тощо) за умови прояву вразливості з використанням експлойта (exploit).

Поява дефектів притаманна всім етапам життєвого циклу системи (об'єкта кіберзахисту). Так, розрізняють дефекти етапу проєктування, реалізації, налаштування при введенні в експлуатацію, супроводженні, завершення роботи. Існування у складі компонентів системи хоча б одного дефекту, який дозволяє здійснити негативний технічний ефект, робить її вразливою. Своєчасний пошук та усунення дефектів зменшує ймовірність компрометації системи від використання, принаймні, відомих експлойтів.

Доцільність, раціональність та обґрунтованість рішень з пошуку дефектів базується на застосуванні апробованих джерел експертного досвіду світового рівня. Так, Adversarial Tactics, Techniques & Common Knowledge визначають системи класу Server-side Web Application привабливим для хакерів об'єктом атак з найбільшою середньою кількістю потенційних дефектів у складі принаймні трьох компонентів: Web Server, Web Application Server, DBMS Server. Для з'ясування розподілу дефектів Server-side Web Application за характерними класами необхідно скористатися Open Worldwide Application Security Project, для детального ознайомлення з ними – Common Weakness Enumeration. Застосування National Vulnerability Database та Common Vulnerabilities and Exposures доповнює оцінку знайденого дефекту.

Ключові слова: *кіберзахист, компрометація системи, оцінка вразливостей*

T. Tereshchenko, V. Ostapchuk, P. Khusainov, Y. Chernysh Assessment of vulnerability of Server-side Web Application and prevention of their manifestation.

The vulnerability of the system (technological, communication) of the cyber protection object to negative technical impact is always based on the exploitation of its defects. Defect is a property of software, hardware, software-hardware or system components, which under certain conditions can lead to vulnerability. Attack – an attempt to exploit a defect(s) of the system (object of cyber protection) for the implementation of a negative technical effect (unauthorized reading/writing of data, incorrect operation, circumvention of protective mechanisms, abnormal consumption of resources; execution of imposed commands, etc.) under the manifestation of vulnerability using an exploit (exploit).

The appearance of defects is inherent in all stages of the system (cyber protection object) life cycle. Thus, defects are distinguished at the design stage, implementation, adjustment during commissioning, maintenance, and completion of work. The existence of at least one defect in the components of the system, which allows a negative technical effect, makes it vulnerable. Timely detection and elimination of defects reduces the probability of system compromise using at least known exploits.

The expediency, rationality and reasonableness of solutions for finding defects is based on the application of proven sources of world-class expert experience. Thus, Adversarial Tactics, Techniques & Common Knowledge defines Server-side Web Application class systems as an attractive attack target for hackers with the highest average number of potential defects in at least three components: Web Server, Web Application Server, DBMS Server. To find out the distribution of Server-side Web Application defects by characteristic classes, it is necessary to use the Open Worldwide Application Security Project, for a detailed study of them - Common Weakness Enumeration. The application of the National Vulnerability Database and Common Vulnerabilities and Exposures complements the evaluation of the found defect.

Keywords: *cyber protection, system compromise, vulnerability assessment.*

Постановка завдання. Суб'єкти забезпечення кібербезпеки у межах своєї компетенції повинні забезпечувати періодичний аудит інформаційної безпеки систем (об'єктів

кіберзахисту) із запровадженням кращих світових практик та міжнародних стандартів. Базуючись на принципах пропорційності, адекватності і пріоритетності запобіжних заходів кіберзахисту реальним та потенційним ризикам аудит інформаційної безпеки, перш за все, спрямований на запобігання порушень у системі (об'єкта кіберзахисту): режиму сталого, надійного та нормального функціонування служб (функцій, компонентів); конфіденційності, цілісності та доступності інформаційних ресурсів [1–3].

Обов'язковою складовою аудиту інформаційної безпеки є пошук та виявлення потенційної вразливості (vulnerability) системи (технологічної, комунікаційної) об'єкта кіберзахисту. Вразливість системи – властивість системи, через використання якої створюється загроза для її безпеки, порушується сталий, надійний та штатний режим функціонування системи, здійснюється несанкціоноване втручання в її роботу, створюється загроза для безпеки (захищеності) електронних інформаційних ресурсів, конфіденційності, цілісності, доступності таких ресурсів. Дослідник потенційної вразливості (далі – дослідник) – фізична або юридична особа, яка здійснює пошук потенційної вразливості системи.

Організація пошуку потенційної вразливості системи здійснюється її власником. Пошук потенційної вразливості системи здійснюється на підставі публічної пропозиції. У публічній пропозиції, зокрема, визначається:

інформація про систему, пошук потенційної вразливості якої здійснюється;

дії дослідника щодо системи, які йому заборонено проводити;

порядок надання дослідником звіту, вимоги до його підготовки, форми.

Після завершення пошуку потенційної вразливості системи дослідник повідомляє про результати власнику системи або координатору згідно з умовами публічної пропозиції та подає йому звіт. За результатами перевірки звіту власник системи оцінює можливі наслідки використання вразливості системи для її безпеки, порушення сталого, надійного та штатного режиму її функціонування, здійснення несанкціонованого втручання в її роботу, створення загрози для безпеки (захищеності) електронних інформаційних ресурсів, конфіденційності, цілісності, доступності таких ресурсів (далі – наслідки вразливості системи) та приймає рішення щодо внесення або невнесення змін до системи.

Звіт про вразливість повинен принаймні містити опис дефекту (дефектів) та негативного технічного ефекту (negative technical impact) для системи. Вразливість системи до здійснення негативного технічного ефекту завжди базується на експлуатації її дефектів. Дефект (weakness) – властивість програмних, апаратних, програмно-апаратних або системних компонентів, які при певних умовах можуть призвести до вразливості. Атака (attack) – спроба експлуатації дефекту (дефектів) системи (об'єкта кіберзахисту) для здійснення негативного технічного ефекту за умови прояву вразливості на основі використання експлойта (exploit). Викладення дефекту (дефектів), негативного технічного ефекту спрямоване на формування технічного та контекстного розуміння вразливості.

Поява дефектів притаманна всім етапам життєвого циклу системи (об'єкта кіберзахисту). Так, розрізняють дефекти етапу проєктування, реалізації та налаштування (при введенні в експлуатацію, супроводженні, завершенні функціонування). Існування у складі компонентів системи хоча б одного дефекту, який дозволяє здійснити негативний технічний ефект, робить її вразливою. Своєчасний пошук та усунення дефектів зменшує ймовірність компрометації системи з використання, принаймні, відомих.

Для з'ясування тактик, технік, процедур (роботи експлойтів) компрометації систем (об'єктів кіберзахисту) слід звернутися до класифікаторів Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) та Common Attack Pattern Enumerations and Classifications (CAPEC), які сформовані на основі розслідування відомих кіберінцидентів/кібертатак. Компрометація системи – порушення сталого, надійного та штатного режиму функціонування систем та/або порушення конфіденційності, цілісності, доступності інформації (інших

властивостей інформації), що обробляється в цих системах. Тактики, техніки, процедури – ієрархічна модель опису поведінки та дій актора (користувача, адміністратора, зловмисника) щодо будь-якої діяльності в системах/мережах, а також спроб діяльності з метою впливу на ці системи [4–7].

Методичним базисом та обов’язковим інструментарієм діяльності дослідника для аналізу ознак вразливості, оцінок, наслідків (прояв негативного технічного ефекту) тощо є Common Vulnerabilities and Exposures (CVE), Common Weakness Enumeration (CWE) [8–9].

Аналіз публікацій про вразливості та дефекти систем. Станом на 06.05.2024 облікована 233 151 вразливість продуктів та технологічних рішень у сфері інформаційних технологій. Розрізняють 11 категорій вразливості, які уособлюють технологічну основу здійснення негативного технічного ефекту. Вразливість є наслідком відсутності або неправильності роботи (недостатності, помилковості, некоректності) процедур контролю (запобігання) непередбаченого використання невід’ємних функцій компонентів системи:

Overflow – перевищення ємності місця їх розташування даних;

Memory Corruption – пошкодження пам’яті;

Sql Injection – поява в запитах мовою Structured Query Language зовнішніх даних;

Cross-Site Scripting (XSS) – обробка компрометованого гіперпосилання;

Directory Traversal – зміна робочої локації (каталогу) у файловій системі;

File Inclusion – нав’язування обробки файлу;

Cross-Site Request Forgery (CSRF) – підміна суб’єкта доступу за запитом;

XML External Entity (XXE) – обробка зовнішніх об’єктів eXtensible Markup Language;

Server-side Request Forgery (SSRF) – підміна суб’єкта доступу до довіреного сервісу;

Open Redirect – модифікація шляху доступу;

Input Validation – порушення фільтрації вхідних даних.

Середній процент загальної ваги категорії вразливості за 10 років: Overflow – 20 %; Memory Corruption – 20 %; Sql Injection – 9 %; Cross-Site Scripting – 25 %; Directory Traversal – 5 %; File Inclusion – 1,5 %; Cross-Site Request Forgery – 6 %; XML External Entity – 1,5 %; Server-side Request Forgery – 1,5 %; Open Redirect – 1,5 %; Input Validation – 9 %.

Окрім опису технологічної основи здійснення негативного технічного ефекту вразливість має доповнення у вигляді 5 форм доведеної практичної експлуатації:

Code Execution – виконання команд, програм із зовнішніх джерел;

Bypass – подолання захисних механізмів;

Privilege Escalation – розширення повноважень;

Denial of Service – порушення (припинення) функціонування;

Information Leak – порушення конфіденційності інформації.

За останні 10 років обліковано 69 833 нових вразливості, з них Code Execution – 19 052; Bypass – 7 114; Privilege Escalation – 9 920; Denial of Service – 22 644; Information Leak – 11 103.

Розрізняють 938 типових дефектів програмних, апаратних та програмно-апаратних компонентів систем (об’єкта кіберзахисту) за 374 категоріями, які були притаманні протягом 17 виділених фаз життєвого циклу системи (проєктування – 3, реалізація – 6, введенні в експлуатацію – 2, супроводження – 5, звершення роботи – 1) [7–9].

Формулювання мети статті. Оприлюднити результати проведеного дослідження категорій вразливості Server-side Web Application на основі оцінок за визначеними показниками та обґрунтовану дорожню карту заходів для запобігання їх прояву.

Виклад основного матеріалу. Поняття Web Application (далі також – Web-застосування) є сучасною практичною реалізацією ідеї, принципів та технологій доступу користувача комп’ютера до інформації, яка оформлена у вигляді документа мовою Hyper Text Markup Language (далі – HTML-документ) з використанням Hyper Text Transfer Protocol (далі – HTTP-протокол). Взаємодія двох програм за HTTP-протоколом забезпечує доступ до

HTML-документа шляхом передачі вмісту відповідного файлу. Абстрагуючись від несуттєвих особливостей, алгоритм роботи однієї з взаємодіючих програм передбачає здійснення запиту на одержання файлу HTML-документа (далі – HTTP-клієнт), інша програма – HTTP-сервер – здійснює його відправку (за запитом) зі своєї файлової системи (рис. 1). Головною особливістю мови HTML-документа є можливість використання, так званих, гіперпосилань на інші HTML-документи за місцем розташування та іменами відповідних їх файлів. Якщо файли, на які вказують гіперпосилання, знаходяться у тій самій файловій системі, то говорять про його локальне розташування відносно файлу початкового HTML-документа з іменем index.html, якщо у файловій системі іншого комп’ютера – зовнішнє. Унікальне та однозначне іменування файлів (у файловій системі) HTML-документів здійснюється у формі Uniform Resource Locator (далі – URL-адреса) завдяки включення доменного імені або мережевої адреси розташування комп’ютера.

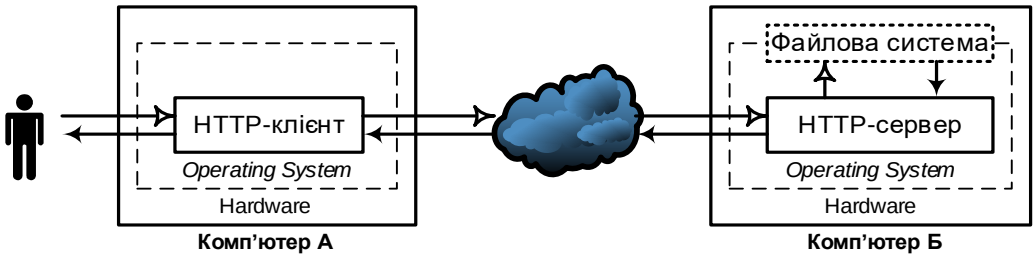

Рис. 1. Основні компоненти взаємодії комп’ютерів за Hyper Text Transfer Protocol

Web-застосування використовують набір серверних сценаріїв (Java, C#, Ruby, PHP тощо) і клієнтських сценаріїв (HTML, JavaScript тощо) для виконання програми. Робота вебдодатка залежить від його архітектури, яка включає апаратне та програмне забезпечення, яке виконує такі завдання, як читання запиту, а також пошук, збір і відображення необхідних даних. Архітектура включає різні пристрої, Web-браузери та зовнішні служби. Розрізняють архітектурні рівні клієнта або презентації, бізнес-логіки, бази даних. Базуючись на мовах програмування, які підтримуються браузером, наприклад, JavaScript, HTML і CSS, у поєднанні з іншими мовами програмування, такими як SQL, для доступу до даних із баз даних (рис. 2).

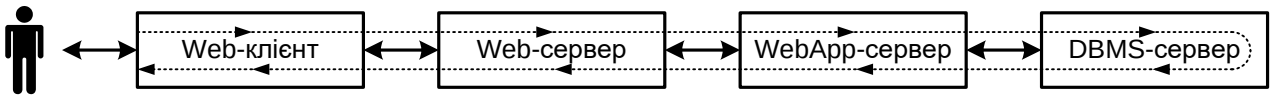

Рис. 2. Комплекс компонентів реалізації Web-застосування

Web-сервіс – це програма або програмне забезпечення, яке розгортається через Інтернет. Він використовує стандартний протокол обміну повідомленнями (наприклад, SOAP) для забезпечення зв’язку між додатками, розробленими на різних платформах. Наприклад, сервіси на основі Java можуть взаємодіяти з PHP-додатками. Ці вебдодатки інтегруються з SOAP, UDDI, WSDL і REST через мережу (рис. 3).

Архітектура Web-сервісу описує взаємодію між постачальником послуг, запитувачем послуг і реєстром послуг. Ця взаємодія складається з трьох операцій, а саме: публікація, пошук і зв’язування. Всі ці ролі та операції працюють разом над артефактами вебсервісу, відомими як програмні модулі (сервіси) та їхні описи. Постачальники послуг пропонують вебсервіси. Вони розгортають і публікують описи вебсервісів у реєстрі сервісів. Запитувачі знаходять ці описи в реєстрі послуг і використовують їх, щоб зв’язатися з постачальником вебсервісу і викликати реалізацію вебсервісу. В архітектурі вебсервісу розрізняють ServiceProvider (постачальник послуг); ServiceRequester (запитувач послуг); ServiceRegistry (реєстр послуг). Logic tier містить код, який зчитує дані з браузера та повертає результати. Рівень бізнес-логіки

включає функціональну логіку Web-застосування, яка реалізована за допомогою таких технологій, як .NET, Java та «проміжне програмне забезпечення». Він визначає потік даних, відповідно до якого розробник будує додаток за допомогою мов програмування. Він зберігає дані програми та інтегрує застарілі програми з найновішими функціями програми. Серверу потрібен певний протокол для доступу до запитуваних користувачем даних із його бази даних. Цей рівень містить програмне забезпечення та визначає кроки для пошуку та отримання даних. Data tier складається з сервера бази даних, який надає виробничі дані організації в структурованій формі (наприклад, MS SQL Server, сервер MySQL).

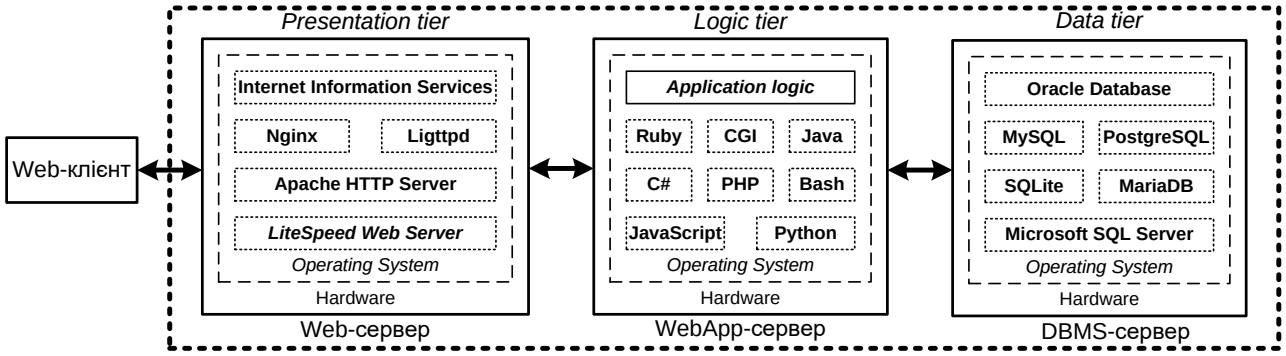

Рис. 3. Компоненти Server-Side Web Application

Існують невід'ємні ризики безпеки, пов'язані з вебсерверами, локальними мережами (LAN), у яких розміщені вебсайти, і кінцевими користувачами, які отримують доступ до цих вебсайтів за допомогою браузерів. Уразливі місця неправильно налаштованих вебсерверів можуть бути пов'язані з конфігурацією, програмами, файлами, сценаріями або вебсторінками. Зловмисник шукає такі вразливі вебсервери для здійснення атак. Неправильна конфігурація вебсервера надає зловмисникові шлях до цільової мережі організації. Неправильна конфігурація вебсервера відноситься до слабких місць конфігурації вебінфраструктури, які можна використати для запуску різних атак на вебсервери, таких як обхід каталогу, вторгнення на сервер і крадіжка даних. Адміністратори, які неправильно налаштовують вебсервери, можуть залишити серйозні лазівки у вебсервері, таким чином надаючи зловмиснику можливість використати неправильно налаштований вебсервер, щоб поставити під загрозу його безпеку та отримати конфіденційну інформацію.

Неправильна конфігурація відноситься до слабких місць конфігурації, які можна використати для запуску різних атак, таких як обхід каталогу, вторгнення на сервер і крадіжка даних. Типові недоліки конфігурації:

детальні повідомлення про налагодження/помилки;
анонімні або стандартні користувачі/паролі;
файли конфігурації та сценарії за замовчуванням;
функції віддаленого адміністрування;
увімкнено непотрібні служби;
неправильно налаштовані сертифікати;
неправильно налаштовані параметри шифрування;
сертифікати за замовчуванням;
використання самопідписаних сертифікатів;
неналежні дозволи доступу для файлів і каталогів;
увімкнено непотрібні служби,
увімкнено керування вмістом;
безпека суперечить вимогам компанії до простоти використання;
відсутність належної політики безпеки, процедур і обслуговування;

неналежна автентифікація за допомогою зовнішніх систем;

облікові записи за замовчуванням із паролями за замовчуванням або без них;

непотрібні файли за замовчуванням, резервні копії або зразки;

помилки в серверному програмному забезпеченні;

неправильно налаштована реєстрація подій;

адміністративні функції увімкнені або доступні на вебсерверах.

Оцінки категорій вразливості Server-side Web Application згідно з The Open Worldwide Application Security Project(OWASP) [11]:

CWEs Mapped – кількість типових дефектів, що мали місце для прояву вразливості;

Max Incidence Rate, Avg Incidence Rate – максимальний та середній процент існуючих Web Application-продуктів, які мають дефекти з переліку CWE для цієї категорії вразливості;

Max Coverage, Avg Coverage – максимальний та середній процент Web Application-продуктів, які мають хоча б один дефект з переліку CWE для цієї категорії вразливості відносно всіх Web Application-продуктів всіх організацій;

Total Occurrences – загальна кількість ідентифікованих Web Application-продуктів, які мають хоча б один дефект з переліку CWE для цієї категорії вразливості;

Total CVEs – загальна кількість CVE, для яких є характерним згадування хоча б одного з переліку CWE для цієї категорії вразливості

Broken Access Control. Недостатність (недосконаліть, некоректність, неправильна робота, відсутність) процедур контролю доступу може призвести до:

порушення принципу надання найменших привілеїв за замовчуванням;

обходу процедур перевірки повноважень;

несанкціонованого розширення повноважень, використання маркерів доступу;

несанкціонованого використання функцій інтерфейсу прикладного програмування (Application Programming Interface, API), посилань на об’єкти.

Оцінки Broken Access Control: CWEs Mapped = 34; Max Incidence Rate = 55,97 %; Avg Incidence Rate = 3,81 %; Max Coverage = 94,55 %; Avg Coverage = 47,72 %; Total Occurrences = 318487; Total CVEs = 19013.

Cryptographic Failures. Недостатність (недосконаліть, некоректність, неправильна робота, відсутність) процедур криптографічного перетворення може призвести до:

передбаченого, за замовчуванням, перемикання на використання незахищених комунікаційних протоколів, відмови від утворення захищеного каналу;

передбаченого, за замовчуванням, перемикання на використання менш стійких криптографічних алгоритмів, ключів, векторів ініціалізації або гам шифру тощо;

відсутності контролю дійсності ключів, цифрових підписів, програм (програмних компонентів) у репозиторіях, у ланцюгах підтвердження дійсності ідентифікаційних сертифікатів тощо;

небажаної (небезпечної, надлишкової) інформативності реакції компонентів при обробці некоректних (неправильних) комбінацій зашифрованих значень.

Оцінки Cryptographic Failures: CWEs Mapped = 29; Max Incidence Rate = 46,44 %; Avg Incidence Rate = 4,49 %; Max Coverage = 79,33 %; Avg Coverage = 34,85 %; Total Occurrences,= 233788; Total CVEs = 3075.

Injection. Недостатність (недосконаліть, некоректність, неправильна робота, відсутність) процедур контролю коректності вхідних даних може призвести до:

обробки небезпечних (некоректних) вхідних даних;

некоректного співставлення та інтерпретації об’єктів;

небажаної (небезпечної, надлишкової) інформативності реакції компонентів при обробці некоректних (неправильних) запитів до системи керування базою даних.

Оцінки Injection: CWEs Mapped = 33; Max Incidence Rate = 19,09 %; Avg Incidence Rate = 3,37 %; Max Coverage = 94,04 %; Avg Coverage = 47,90 %; Total Occurrences = 274228; Total CVEs = 32078.

Insecure Design. Небезпечний дизайн уособлює сукупність порушень моделі циклу безпечної розробки OWASP Software Assurance Maturity Model (SAMM), яка базується на здійсненні систематичного аналізу та усуненні загроз на всіх етапах розробки компонентів на основі використання вже апробованих безпечних з позиції безпеки елементів.

Оцінки Insecure Design: CWEs Mapped = 40; Max Incidence Rate = 29,19 %; Avg Incidence Rate = 3,00 %; Max Coverage = 77,25 %; Avg Coverage = 42,51 %; Total Occurrences = 262407; Total CVEs = 2691.

Security Misconfiguration. Недостатність (недосконаліть, некоректність, помилкове полягання на параметри за замовчуванням) технологічних процедур таких аспектів організації безпеки, як:

використання хмарних сервісів;

запобігання роботі компонентів та доступності облікових записів, які не використовуються, у тому числі тих, що інстальовані (створені) за замовчуванням;

запобігання (ускладнення) одержанню суб’єктом атаки відомостей про архітектуру server-side на основі аналізу відповідей за запити, у тому числі некоректні, недоречні тощо;

контроль та запобігання утворенню незахищених каналів комунікації, застосуванню менш стійких режимів захищених комунікаційних протоколів;

контроль використання актуальних версій та оновлень безпеки програмних компонентів;

систематичний аналіз ризиків.

Оцінки Security Misconfiguration: CWEs Mapped = 20; Max Incidence Rate = 19,84 %; Avg Incidence Rate = 4,51 %; Max Coverage = 89,58 %; Avg Coverage = 44,84 %; Total Occurrences = 208387; Total CVEs = 789.

Vulnerable and Outdated Components. Недостатність (недосконаліть, помилковість) організації технологічних процедур систематичного пошуку вразливостей, контролю використання застарілих (неактуальних) версій, систематичного оновлення реалізації програмних компонентів Web-серверів, WebApp-серверів, DBMS-серверів (операційних систем, системних бібліотек функцій, інтерпретаторів мов програмування тощо).

Оцінки *Vulnerable and Outdated Components*: CWEs Mapped = 3; Max Incidence Rate = 27,96 %; Avg Incidence Rate = 8,77 %; Max Coverage = 51,78 %; Avg Coverage = 22,47 %; Total Occurrences = 30457; Total CVEs = 0.

Identification and Authentication Failures. Недостатність (недосконаліть, некоректність, помилкова впевненість в параметрах за замовчуванням) технологічних процедур ідентифікації та автентифікації може призвести до набуття суб’єктом атаки повноважень в системі на основі, а саме:

підбору автентифікаційного параметрів (одноразових паролів, маркерів, ідентифікаторів тощо) в процесі віддаленої комунікації між обчислювальними процесами Web-клієнта та компонентами Web-сервера, WebApp-сервера, DBMS-сервера, а також в процесі внутрішньої комунікації між ними;

підбору нестійких до угадування паролів користувачів (внаслідок відсутності відповідного контролю при створенні), у тому числі двофакторної автентифікації (через нестійкість до угадування та/або прогнозування можливих значень);

компрометації значення паролю через критичне порушення логіки (при проєктуванні та створенні) процесу відновлення пароля, у тому числі можлива відмова від його використання при певних умовах;

витоку незашифрованих значень паролів або зашифрованих з використанням нестійких алгоритмів хешування паролів;

повторного використання одноразових паролів (маркерів, ідентифікаторів тощо) через критичне порушення логіки (при проєктуванні та створенні) процесу їх видання.

Оцінки Identification and Authentication Failures: CWEs Mapped = 22; Max Incidence Rate = 14,84 %; Avg Incidence Rate = 2,55 %; Max Coverage = 79,51 %; Avg Coverage = 45,72 %; Total Occurrences = 132195; Total CVEs = 3897.

Software and Data Integrity Failures. Недостатність (недосконаліть, некоректність, помилкова впевненість в параметрах за замовчуванням, відсутність) технологічних процедур організації контролю автентичності, цілісності, безпечності даних (включаючи у складі серіалізованих об’єктів, конвеєрів обробки даних) програмного забезпечення, які надходять з репозиторіїв (постачаються на зовнішніх носіях) для інсталяції (оновлення) компонентів Web-сервера, WebApp-сервера, DBMS-сервера.

Оцінки Software and Data Integrity Failures: CWEs Mapped = 10; Max Incidence Rate = 16,67 %; Avg Incidence Rate = 2,65 %; Max Coverage = 75,04 %; Avg Coverage = 45,35 %; Total Occurrences = 47972; Total CVEs = 1152.

Security Logging and Monitoring Failures. Недостатність (недосконаліть, некоректність, помилкова впевненість в параметрах за замовчуванням) технологічних процедур реєстрації, обліку та систематичного аналізу повідомлень про події функціонування Web-сервера, WebApp-сервера, DBMS-сервера, а також введення архівного (резервного) зберігання журнальних файлів. Найбільш суттєві аспекти прояву негативного впливу цієї категорії дефектів:

низька інформативність (прагматична цінність) повідомлень про події через неузгодженість їх форматів (синтаксис, семантика повідомлень) від різнотипних джерел;

недоступність резервних або архівних копій журнальних файлів для аналізу у разі їх цілеспрямованої модифікації (пошкодження, фальсифікація, знищення) суб’єктом атаки;

неможливість застування ретроспективного аналізу архівних журнальних файлів при розслідуванні кіберінцидентів;

потрапляння критичної технологічної інформації, значень параметрів ідентифікації, автентифікації тощо до вмісту повідомлень про події, що є передумовою для її неконтрольованого розповсюдження та витоку.

Оцінки Security Logging and Monitoring Failures: CWEs Mapped = 4; Max Incidence Rate = 19,23 %; Avg Incidence Rate = 6,51 %; Max Coverage = 53,67 %; Avg Coverage = 39,97 %; Total Occurrences = 53615; Total CVEs = 242.

Server-Side Request Forgery. Недостатність (недосконаліть, некоректність, неправильна робота, відсутність) процедур контролю небезпечного (некоректного) вмісту запитів Web-сервера до інших компонентів Server-side Web Application, які формуються на основі запитів Web-клієнта до Web-сервера, а також систематичного аналізу та усунення відповідних дефектів такого класу.

Оцінки Server-Side Request Forgery: CWEs Mapped = 1; Max Incidence Rate = 2,72 %; Avg Incidence Rate = 2,72 %; Max Coverage = 67,72 %; Avg Coverage = 67,72 %; Total Occurrences = 9503; Total CVEs = 387.

Injection. На третій позиції з 94 % визначених продуктів та 274 тис. випадків. Серед найпомітніших загальних вразливостей CWE-79, CWE-89, CWE-73.

Ознаки вразливості до атаки:

дані, що надаються користувачем, не перевіряються, не фільтруються та не очищуються програмою;

динамічні запити або непараметризовані виклики без контекстно-залежного екранування використовуються безпосередньо в інтерпретаторі;

вхідні дані використовуються у параметрах пошуку об’єктно-реляційного відображення для вилучення додаткових, чутливих записів;

вхідні дані використовуються безпосередньо або конкатенуються.

До цього класу атак відноситься як традиційний SQL, так і NoSQL, нав’язування команд операційній системі, об’єктно-реляційне відображення. Концепція ідентична для всіх інтерпретаторів. Аналіз вихідного коду є найкращим методом виявлення вразливості. Наполегливо рекомендується автоматизоване тестування всіх параметрів, заголовків, URL, файлів cookie, вхідних даних.

Щоб запобігти ін’єкціям, потрібно зберігати дані окремо від команд і запитів. Кращим варіантом є використання безпечного API, який повністю уникає використання інтерпретатора, надає параметризований інтерфейс або мігрує до інструментів відображення об’єктно-реляційних зв’язків. Навіть параметризовані збережені процедури можуть спричинити SQL-ін’єкцію. Для будь-яких залишкових динамічних запитів екрануйте спеціальні символи, використовуючи спеціальний синтаксис екранування для цього інтерпретатора. Структури SQL, такі як назви таблиць, стовпців тощо, не можна екранувати, тому назви структур, введені користувачем, є небезпечними. Це поширена проблема у програмах для написання звітів.

Доступність початкового тексту є дуже важливим фактором захисту та запобігання дефектам ін’єкції. Лише менша частина всіх програм компанії/підприємстві розробляється власними силами, а більшість додатків – із зовнішніх джерел. Програми з відкритим кодом дають принаймні можливість виправити проблеми, але програми із закритим кодом потребують іншого підходу до недоліків впровадження.

Недоліки впровадження виникають, коли програма надсилає ненадійні дані інтерпретатору. Недоліки ін’єкцій дуже поширені, особливо в застарілому коді. Вади легко виявити під час вивчення коду, але важче під час тестування. Сканери та фазери можуть допомогти зловмисникам їх знайти.

Залежно від доступності потрібно виконати різні дії, щоб їх виправити. Це завжди найкращий спосіб вирішити проблему в самому вихідному коді або навіть змінити дизайн деяких частин програми. Але якщо вихідний код недоступний або просто неекономно виправляти застаріле програмне забезпечення, лише віртуальне виправлення має сенс.

Найвідомішою формою впровадження є SQL-ін’єкція, коли зловмисник може змінювати існуючі запити до бази даних. Щоб отримати додаткові відомості, перегляньте шпаргалку щодо запобігання впровадження SQL. Атака SQL-ін’єкції полягає у вставці або «ін’єкції» часткового або повного SQL-запиту через дані, що вводяться або передаються від клієнта (браузера) до вебдодатка.

Успішна атака SQL-ін’єкцій може зчитувати конфіденційні дані з бази даних, змінювати дані бази даних (вставити/оновити/видалити), виконати операції адміністрування бази даних, відновити вміст певного файлу, записувати файли у файлову систему, а в деяких випадках видавати команди операційній системі.

Атаки SQL Injection можна розділити на такі три класи:

1. Дані витягуються за допомогою того самого каналу, який використовується для впровадження коду SQL. Це найпростіший вид атаки, у якому отримані дані представлено безпосередньо на вебсторінці програми.

2. Дані отримуються за допомогою іншого каналу (наприклад, електронний лист із результатами запиту створюється та надсилається тестувальнику).

3. Фактичної передачі даних немає, але тестер може реконструювати інформацію, надсилаючи певні запити та спостерігаючи за результатом поведінки сервера бази даних

Усі мови сценаріїв, що використовуються у вебдодатках, мають форму виклику eval, який отримує код під час виконання та виконує його. Якщо код створено з використанням неперевірених і неекранованих введених користувачем кодів, може статися ін’єкція коду, що дозволить зловмиснику підірвати логіку програми та зрештою отримати локальний доступ.

Якщо мова сценаріїв має недолік у коді обробки даних, вектори атаки «введення нульового байта» можуть розгортатись для отримання доступу до інших областей пам’яті, що призводить до успішної атаки. Впровадження команд операційної системи – це техніка, яка використовується через інтерфейс для виконання команд. Маючи можливість виконувати команди ОС, користувач може завантажувати шкідливі програми або навіть отримувати паролі. Впровадженню команд ОС можна запобігти, якщо під час проєктування та розробки програм наголошується на безпеці.

Дорожня карта заходів запобігання прояву вразливості Server-side Web Application:
виправляти та оновлювати програмне забезпечення;
застосовувати всі оновлення, незалежно від їхнього типу, за потребою;
тестувати пакети оновлень перед розгортанням;
здійснювати резервне копіювання;
розробити план повернення до початкового стану після невдалого оновлення;
планувати періодичні оновлення пакетів;
вимкнути невикористані зіставлення розширень сценаріїв;
усунути використання конфігурацій за замовчуванням;
планувати аварійного відновлення для вирішення проблем керування виправленнями;
блокуйте непотрібні порти;
блокувати непотрібні протоколи;
використовувати безпечну автентифікацію;
використовувати тунелювання та шифрування каналів комунікації;
використовувати безпечні протоколи для зв’язку з вебсервером;
ізолювати каталоги сервісів;
видалити непотрібні модулі та розширення програм;
вимкнути облікові записи користувачів за замовчуванням;
видалити користувачів бази даних і збережені процедури;
дотримуватися принципу найменших привілеїв для бази даних;
використовувати безпечні дозволи;
використовувати політики надійних паролів;
здійснювати перевірки стійкості паролів;
реєструвати помилки автентифкації;
надавати процесам найменші привілеї;
створювати мінімальну кількість привілейованих облікових записів;
обмежити доступ адміністратора;
здійснювати реєстрацію подій у зашифрованому вигляді;
вимкнути усі неінтерактивні облікові записи;
вилучити конфіденційну конфігураційну інформацію з байт-коду;
вилучити зіставлення віртуальних каталогів з різними серверами або через мережу;
здійснювати аналіз журналів обліку подій;.
вимкнути обслуговування списків каталогів;
вилучити застосування непотрібних файлів і нецільових розширень;
вимкнути обслуговування певних типів файлів;
забезпечити ізольоване зберігання файлів;
використання ізольованого програмного середовища;
уникайте посилань на всі типи невебфайлів в URL-адресі.

Висновки. Впровадженню SQL найкраще запобігти за допомогою параметризованих запитів. Кращим варіантом є використання безпечного API, який повністю уникає використання інтерпретатора, надає параметризований інтерфейс або мігрує до інструментів

відображення об’єктно-реляційних зв’язків. Зверніть увагу, що багато фреймворків і бібліотек на стороні клієнта пропонують параметризацію запитів на стороні клієнта. Ці бібліотеки часто просто створюють запити з конкатенацією рядків перед надсиланням необроблених запитів на сервер.

Подальші дослідження передбачають розробку розширеної дорожньої карти заходів, необхідної для запобігання дії додаткових вразливостей Server-side Web Application, які неодмінно будуть виявлені в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про основні засади забезпечення кібербезпеки України: Закон України від 05.10.2017 № 2163-VIII. *Офіційний вісник України.* 2017. № 91. С. 91.

2. Про захист інформації в інформаційно-комунікаційних системах: Закон України від 05.07.1994 № 80/94-ВР. *Відомості Верховної Ради України.* 1994. № 31. Ст. 286.

3. Про затвердження Загальних вимог до кіберзахисту об’єктів критичної інфраструктури: постанова Каб. Міністрів України від 19.06.2019 № 518. *Офіційний вісник України.* 2019. 05 лип. (№ 50). С. 53.

4. Про затвердження Порядку пошуку та виявлення потенційної вразливості інформаційних (автоматизованих), електронних комунікаційних, інформаційно-комунікаційних систем, електронних комунікаційних мереж: постанова Каб. Міністрів України від 16.05.2023 № 497. *Офіційний вісник України.* 2023. 02 черв. (№ 52). С. 72.

5. Про затвердження Методичних рекомендацій щодо реагування суб’єктами забезпечення кібербезпеки на різні види подій у кіберпросторі: наказ Адміністрації Державної служби спеціального зв’язку та захисту інформації України від 03.072023 № 570. *GOV.ua.* URL: https://cip.gov.ua/ua/news/nakaz-administraciyi-derzhspeczv-yazku-vid-03-07-2023-570-pro-zatverdzhennya-metodichnikh-rekomendacii-shodo-reaguvannya-sub-yektami-zabezpechennya-kiberbezpeki-na-rizni-vidi-podii-u-kiberprostori.

6. Adversarial Tactics, Techniques & Common Knowledge. URL: https://attack.mitre.org.

7. Common Attack Pattern Enumerations and Classifications. URL: https://capec.mitre.org.

8. Common Vulnerabilities and Exposures. URL: https://cve.mitre.org.

9. Common Weakness Enumeration. URL: https://cwe.mitre.org.

10. Open Worldwide Application Security Project. URL: https://owasp.org.