УДК 004.7

Лещенко Б.С., аспірант,
Єфіменко А.А., к.т.н., доцент
Державний університет «Житомирська політехніка»

ЗМЕНШЕННЯ РОЗМІРУ КОНТЕЙНЕРНИХ ОБРАЗІВ ЯК СТРАТЕГІЯ ЗНИЖЕННЯ ПОВЕРХНІ АТАКИ

Одним з основних переваг використання контейнеризованих образів, порівняно з віртуальними машинами, стало зменшення розмірів образів. Зростання використання контейнерних технологій спровокувало розвиток досліджень, зосереджених на мінімізації розміру образів *Docker* для підвищення ефективності розгортання та зменшення безпекових ризиків.

Мінімізація кількості вбудованих компонентів та бібліотек сприяє звуженню «поверхні атаки», що, відповідно, обмежує потенційний вектор для експлуатації вразливостей.

Утиліта *BusyBox* [1] була одним з найперших рішень для створення надлегких образів. *BusyBox* об'єднує численні базові *Unix*-команди в одному виконуваному файлі. Образ, що містить лише *BusyBox* та мінімальну реалізацію libc, компактний і підходить для запуску простих бінарних застосунків з мінімальною кількістю залежностей.

Наступним кроком стала поява *Alpine Linux*. *Alpine* є «легким» дистрибутивом, який з самого початку був спроектований для мінімізації свого розміру. *Alpine* використовує *musl-libc* і, той же, набір утиліт *BusyBox*, та власний пакетний менеджер. Однак, слід зауважити, що стандартний Alpine-образ може містити CVE, які не одразу фіксуються традиційними сканерами.

Наступний етап в еволюції безпеки контейнерних образів привніс концепцію так званих *«distroless»* [2] образів, розроблених компанією Google. Distroless образи не тільки виключають непотрібні утиліти, як попередні приклади, а також застосовують радикальне скорочення компонентів, включно з інтерпретатором командного рядка (*shell*) та менеджером пакетів.

Появу декларативних інструментів для побудови мінімальних образів можна назвати новим етапом розвитку контейнеризації. Наприклад, утиліта *apko* від *Chainguard* приймає конфігураційний *YAML*-файл з переліком пакетів та автоматично вирішує залежності, формуючи кінцевий образ.

Ubuntu Chiseled реалізує концепцію package slicing. Package slicing деконструкція Debian пакетів на логічні підмножини залежностей, якими можна легко керувати для створення образів.

У 2025 році також розширено увагу до мінімальних образів на рівні інструментів безпеки. Зокрема, *Amazon* оголосила [3], що *Amazon Inspector* навчають сканувати *distroless* та *Chainguard* образи нарівні з іншими образами. Крім того, нові версії *Docker* та *BuildKit* пропонують додаткові можливості [4]: підтримку стиснення шарів через Zstd/estargz, оптимізацію шарів контейнеру та інші поліпшення, що сприяють створенню ще компактніших образів. Хоч ці новації орієнтовані радше на продуктивність, вони також підвищують безпеку за рахунок зменшення розміру і кращої відтворюваності збірки (в невеликих образах простіше відстежувати зміни).

```
docker images | sort -k7 -h -r
debian                                        latest    8f6a88feef3e   8 days ago      206MB
registry.access.redhat.com/ubi9/ubi-minimal   latest    61d5ad475048   8 days ago      153MB
ubuntu                                        latest    c35e29c94501   5 weeks ago     139MB
debian                                        12-slim   b4aa902587c2   8 days ago      136MB
alpine                                        latest    4b7ce07002c6   6 weeks ago     26.1MB
busybox                                       latest    e3652a00a2fa   14 months ago   13MB
cgr.dev/chainguard/static                     latest    d44809cee093   2 weeks ago     6.41MB
gcr.io/distroless/static                      nonroot   e8a4044e0b4a   N/A             6.23MB
REPOSITORY                                    TAG       IMAGE ID       CREATED         SIZE
```

Рисунок 1 - Порівняння розмірів різних базових образів

На рисунку 1 зображено градацію контейнерних образів за їхнім розміром, показуючи еволюцію від повнофункціональних дистрибутивів до максимально спрощених середовищ.

Таким чином, еволюція підходів до безпеки мінімальних контейнерних образів пройшла шлях від повноцінних дистрибутивів до радикально мінімізованих рішень із суворим контролем вхідних/вихідних даних. Ці сучасні підходи базуються на ідеї обмеження кількості компонентів та наданні розробникам інструментів для декларативного контролю складу образів.

Список використаних джерел:

1. Docker. Busybox. URL: https://hub.docker.com/_/busybox.
2. Google Container Tools. Distroless images. URL: https://github.com/GoogleContainerTools/distroless.
3. Docker. Docker’s developer innovation: unveiling performance milestones. URL: https://www.docker.com/blog/dockers-developer-innovation-unveiling-performance-milestones/.
4. AWS. Amazon inspector enhances container security by mapping amazon ECR images to running containers. URL: https://aws.amazon.com/blogs/aws/amazon-inspector-enhances-container-security-by-mapping-amazon-ecr-images-to-running-containers/.