Dokumentacja wstępna projektu z BSO
Temat: System skanowania urządzeń sieciowych

Miłosz Kutyła, Jakub Ossowski

Politechnika Warszawska

21 kwietnia 2023

Oświadczenie

Niniejszy dokument to dokumentacja wstępna z projektu w ramach przedmiotu BSO. Oświadczamy, że ta praca, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu BSO, została wykonana przez nas samodzielnie.

1 Cel projektu

Celem projektu jest przygotowanie narzędzia dla niezaawansowanego użytkownika, które w sposób automatyczny będzie okresowo generowało raporty bezpieczeństwa. Rozwiązanie ma korzystać z popularnych narzędzi do wykonywania testów penetracyjnych. Gotowe rozwiązanie powinno zostać zamknięte do postaci kontenera wirtualizacyjnego (np. Docker) lub maszyny wirtualnej (KVM, VMware itd.). Kontener powinien zostać opublikowany na koncie przedmiotu w serwisie DockerHub.

2 Interpretacja zadania

W dobie zwiększającej się liczby zagrożeń sieciowych i rosnącej na ich temat świadomości, skanowanie sieci, wykrywanie podatności i testy penetracyjne stały się istotnym elementem utrzymywania sieci. Konieczne jest stosowanie narzędzi okresowo badających sieć i generujących raporty bezpieczeństwa. System skanowania badanie utrzymywanej sieci dzieli na:

- rekonesans: odkrywanie sieci tj. wykrycie hostów, identyfikowanie portów,
- enumerację: listowanie usług dostępnych za pośrednictwem danego portu,
- ocenę podatności: poszukiwanie luk bezpieczeństwa i ich ocena.

Dwa pierwsze etapy agreguje się do tzw. skanowania sieci. Powyższą listę można dodatkowo rozwinąć o próbę wykorzystania wykrytych podatności - wtedy mówimy o testach penetracyjnych. Na podstawie informacji zdobytych przez skaner możliwe jest generowanie raportów, a następnie naprawa wykrytych luk bezpieczeństwa.

3 Skanowanie sieci

Skanowanie jest jednym z najważniejszych elementów utrzymania sieci. Pozwala zapewnić bezpieczeństwo użytkowników i jest jedną z podstawowych metod wczesnego wykrywania potencjalnych zagrożeń. Z tego powodu ważne jest, aby wykorzystywane do tego celu oprogramowanie wykonywało je w sposób pełny i dokładny. Skanowanie sieci realizuje się w czterech komplementarnych etapach:

- odkrywanie hostów: wysyłanie wiadomości na każdy adres IP w sieci. Jeśli jest pod nim aktywny host, to na nią odpowie.
- odkrywanie portów: wysyłanie wiadomości na porty aktywnych hostów w celu sprawdzenia ich aktywności.

- odkrywanie usług i ich wersji: wysyłanie spreparowanych wiadomości na odkryte aktywne porty w celu wygenerowania przez nie odpowiedzi, na podstawie których można określić typ oraz wersję działających na nich usług.
- odkrywanie systemu operacyjnego: wysyłanie spreparowanych wiadomości do hosta w celu wygenerowania konkretnych typów odpowiedzi, na podstawie których można określić typ systemu operacyjnego, który na nim działa.

3.1 Odkrywanie hostów

Pierwszy z wykonywanych kroków podczas skanowania sieci. Ma na celu zredukowanie zakresu adresów IP, wykorzystywanych w późniejszych etapach skanowania. Podstawową stosowaną techniką jest wysyłanie pakietów `ICMP echo request` na wszystkie możliwe adresy i oczekiwanie na odpowiedź. W przypadku otrzymania odpowiedzi host jest oznaczany jako aktywny. Podobną, bardziej dyskretną techniką jest połowiczne otwieranie połączeń (ang. *half-open*). Opiera się na nienawiązywaniu pełnego połączenia TCP, a jedynie na oczekiwaniu na pakiet SYN/ACK lub RST po inicjalizacji połączenia. Po odebraniu pakietu połączenie jest zamykane przez skaner. Nienawiązanie pełnego połączenia skutkuje brakiem odnotowania go w logach skanowanego hosta.

3.2 Odkrywanie portów

Kolejny krok w procesie skanowania sieci ma na celu wykrycie otwartych portów na aktywnych hostach. Każdy port to potencjalny wektor ataku, dlatego poprawna klasyfikacja portów jest kluczowa dla zapewnienia bezpieczeństwa sieci. Przeskanowane porty są najczęściej oznaczane jako otwarte/zamknięte i filtrowane/niefiltrowane. Otwartość oznacza, że na porcie jest wystawiona jakaś aplikacja lub usługa, która aktywnie akceptuje połączenia. Zamknięty port odpowiada na zapytania, ale nie posiada aplikacji będącej w stanie obsłużyć połączenie. Port filtrowany to port, z którym komunikacja jest uniemożliwiona przez różnego rodzaju mechanizmy przykładowo reguły routera lub firewall.

3.3 Odkrywanie usług i ich wersji

Polega na wysyłaniu spreparowanych zapytań charakterystycznych dla określonych serwisów, usług itp. za pomocą bazy danych typowych zapytań. Następnie na podstawie odpowiedzi określa się aktywną usługę na danym porcie, a docelowo także jej wersję (nie zawsze jest to osiągalne). Poznanie dokładnej wersji wystawionej do sieci usługi umożliwia atakującemu znalezienie jej znanych podatności, a ostatecznie wykorzystanie ich i zagrożenie bezpieczeństwu utrzymywanej infrastruktury. W analogiczny sposób możliwe jest wykrycie typu oraz wersji systemu operacyjnego działającego na hoście.

4 Wykrywanie skanowania

Budując dobry skaner portów trzeba mieć świadomość istnienia mechanizmów, które mogą zablokować lub ograniczyć jego działanie. Jednym z nich jest zwracanie szczególnej uwagi na duże ilości zapytań przychodzących na dany port. W zależności od rozważanej sieci może być to realizowane przez np. blokowanie portu po otrzymaniu więcej niż N pakietów SYN, ACK, FIN lub pakietów TCP z wyróżniającymi się flagami (niewystępującymi często w danej sieci) w ciągu T sekund. Można stosować również blokowanie wiadomości `ICMP ECHO` przychodzących do sieci przez firewall lub router.

4.1 Sposoby ominięcia zabezpieczeń

Ponieważ większość sieci wykrywa i blokuje próby skanowania, konieczne jest zastosowanie technik ukrycia skanowania ogólnie określanych mianem *evasion and spoofing*, czyli unikania i podszywania. Mogą być skuteczne w omijaniu firewalli czy systemów wykrywania i zapobiegania włamaniom IDP/IPS - są one często wykonywane przez aktorów atakujących daną sieć. Do takich technik można zaliczyć:

- powolne skanowanie: pozwala na ominięcie wykrycia skanowania przez IDS, które blokują duże liczby połączeń przychodzących w krótkich odstępach czasu. Konsekwencją takiego skanowania jest zdecydowanie wolniejszy wynik, ponie-

waż ogranicza się liczbę hostów i portów skanowanych w danym czasie.
- fragmentacja: polega na rozbiciu wiadomości TCP wysyłanych przez skaner na fragmenty, co może pozwolić na uniknięcie wykrycia.
- podszywanie się i wabiki: skaner może podawać się za adres IP, którym tak naprawdę nie jest, a do sieci dodać tzw. wabiki, które dołączają się do skanowania. Dzięki temu w sieci pojawia się wiele hostów o cechach podobnych do skanera, przez co ciężej jest go wykryć.
- selekcja portów: w celu uniknięcia wykrycia przez firewall jako porty źródłowe można użyć te, które są przez niego przepuszczane.

5 Wykrywanie podatności

Po odkryciu podstawowych informacji na temat sieci skaner może przystąpić do fazy wykrywania luk bezpieczeństwa. W tym celu konieczne jest dostarczenie mu aktualnej bazy podatności, której mógłby użyć przy określaniu potencjalnych wektorów ataku na danego hosta. Dla każdej potencjalnej podatności skaner może zwrócić dwa wyniki:

- pozytywny: oznaczający, że host może być podatny na badany wektor ataku. Należy wziąć pod uwagę możliwość wystąpienia wyników fałszywie pozytywnych.
- negatywny: dana podatność nie może wystąpić u badanego hosta.

Wykryte podatności należy następnie sklasyfikować od najbardziej do najmniej niebezpiecznej. Zapewnia to możliwość ustalenia kolejności naprawy luk bezpieczeństwa. Jednym ze sposobów oceny podatności jest tak zwana miara CVSS, czyli Common Vulnerability Scoring System. Jest ona stosowana w wielu rozwiązaniach takich jak np. OpenVAS. W zależności od typu podatności, czyli jej złożoności, wymagań, powtarzalności, a także ingerencji w triadę CIA danej usługi (poufność, integralność, dostępność), podatność jest klasyfikowana w skali od 0 (najmniej poważna) do 10,0 (najbardziej poważna).

6 Projekt naszego rozwiązania

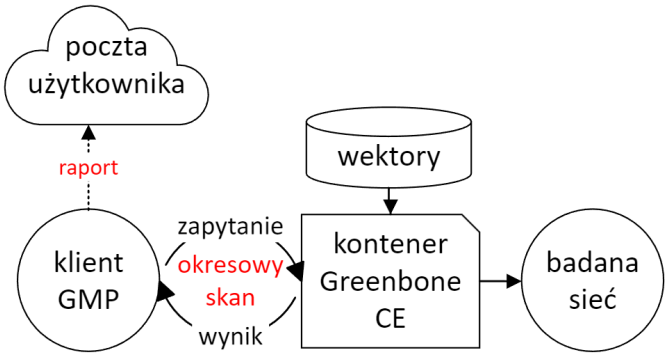
Do zaimplementowania wszystkich opisanych powyżej operacji i technik wykorzystamy rozwiązanie open source - Greenbone Community Edition (22.4). Do jego najważniejszych elementów należą:

- GSA: aplikacja sieciowa do obsługi Greenbone CE i wyświetlania wyników.
- GVMD: deamon, menadżer skanera podatności, umożliwia konfigurację, zwracanie wyników.
- skaner OpenVAS: służący do odkrywania sieci i wykrywania podatności.
- skaner Notus: tzw. skaner lokalny, porównuje podatne wersje aplikacji z tymi zainstalowanymi.
- VTs (NVTs): skrypty wykrywające podatności na zdalnym hoście.

Wybraliśmy Greenbone CE ze względu na:

- ułatwioną możliwość konteneryzacji,
- bogate Python API umożliwiające konfigurację skanera. Komunikacja z GVMD przebiega przy pomocy protokołu GMP.
- dostępność dużej bazy potencjalnych wektorów ataku używanych w trakcie skanowania.

Dodatkowo Greenbone udostępnia przykładowe skrypty pokazujące w jaki sposób można nawiązywać interakcję z GVDM. W szczególności możliwe jest zwracanie wyników skanowania, które następnie możemy samodzielnie wysłać na pocztę użytkownika. Nasze rozwiązanie możemy zatem wstępnie zaplanować. Jego schemat przedstawia rysunek 1.



Rys. 1: Schemat rozwiązania

Zobrazowany na schemacie moduł klienta GMP, który zaimplementujemy w Pythonie, powinien być konfigurowalny przez:

- podanie adresu e-mail, na który wysyłane mają być raporty.
- podanie okresu, co który mają być wykonywane skanowania.

Takiego klienta potem skonteneryzujemy. Następnie dodamy możliwość jego pobrania poprzez plik instalacyjny Greenbone, który przekażemy użytkownikowi końcowemu. Ingerencję użytkownika, z założenia niezaawansowanego, w działanie skanera ograniczymy do jednorazowego wprowadzenia wymaganych danych konfiguracyjnych do naszego modułu. Plik `config` powinien być zatem przechowywany lokalnie, na dysku użytkownika. Wiąże się to z możliwymi zmianami kontenera w trakcie utrzymywania naszego rozwiązania. Zawartość pliku `config` następnie dołączymy do uruchamianego kontenera jako parametr, dzięki czemu użytkownik nie będzie miał problemu z uruchomieniem skanera po instalacji/aktualizacji kontenera. Możemy zatem wstępnie określić strukturę systemu plików naszego rozwiązania:

- $HOME/greenbone-community-container: folder będzie zawierać pliki samego skanera, jego zawartość (pobrane kontenery) definiuje plik `docker-compose-22.4.yml`.
- $HOME/gmp-client-container: folder będzie zawierać nasze rozwiązanie i plik `config`.

Samą instalację ograniczymy zatem do przekazania użytkownikowi pliku wykonywalnego, który po uruchomieniu dokona za niego wszelkie potrzebne operacje. Plik instalacyjny (z uruchomieniem) w swojej docelowej wersji będzie miał następującą postać:

```
#!/bin/sh
BSO=$HOME/gmp-client-container
GBONE=$HOME/greenbone-community-container
CMPS="docker-compose-22.4.yml"
mkdir -p $BSO && cd $BSO
# pobranie kontenera
# wczytanie i zapis konfiguracji
# uruchomienie
cd ..

mkdir -p $GBONE && cd $GBONE
curl -f -O https://greenbone.github.io/
 ↪  docs/latest/_static/$CMPS
docker-compose -f $GBONE/$CMPS -p
 ↪  greenbone-community-edition pull
docker-compose -f $GBONE/$CMPS -p
 ↪  greenbone-community-edition up -d

read -s -p "Haslo dla konta admin: "
 ↪  password
docker-compose -f $GBONE/$CMPS -p
 ↪  greenbone-community-edition \
    exec -u gvmd gvmd gvmd --user=admin
     ↪  --new-password=$password
```

Część dotycząca pobrania i konfiguracji kontenerów Greenbone pochodzi z oficjalnej strony rozwiązania [źródło]. Użytkownikowi końcowemu przekażemy dodatkowo osobny plik uruchamiający pobrane wcześniej kontenery.

Bibliografia

Arkin, Ofir. "Network scanning techniques." *Publicom Communications Solutions*, 1999.

Black, Paul E, Elizabeth Fong, Vadim Okun, Romain Gaucher, et al. "Software assurance tools: Web application security scanner functional specification version 1.0." *Special Publication 500–269*, 2008.

Chauhan, Ajay Singh. *Practical Network Scanning: Capture network vulnerabilities using standard tools such as Nmap and Nessus*. 130–133, 150–155. Packt Publishing Ltd, 2018.

Orebaugh, Angela, and Becky Pinkard. *Nmap in the enterprise: your guide to network scanning*. Elsevier, 2011.

Rahalkar, Sagar, and Sagar Rahalkar. "OpenVAS." *Quick Start Guide to Penetration Testing: With NMAP, OpenVAS and Metasploit*, 2019, 47–71.