Politechnika Wrocławska
Wydział Elektroniki

Kierunek: Cyberbezpieczeństwo

Ochrona centrów danych

Lab 3

Politechnika Wrocławska

Anna Płecha
Aneta Prządka
Krzysztof Bocian

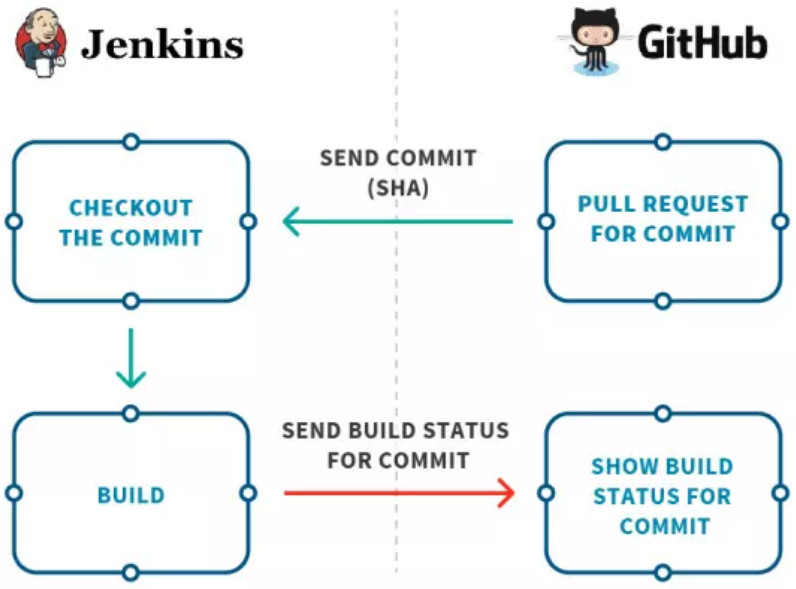
Wrocław, 2021

Spis treści

1.Wstęp

Celem trzeciego laboratorium była dokumentacja infrastruktury z laboratorium 1 i laboratorium 2 oraz zdefiniowanie i kategoryzacja zagrożeń tej infrastruktury. Całość została postawiona na maszynie wirtualnej za pomocą środowiska Oracle VM Virtual Box Manager. Systemem operacyjnym maszyny jest Ubuntu 64-bitowe, zainstalowane w wersji 18.04.6. Właściwościami maszyny jest 4096 MB pamięci RAM, 10 GB pamięci oraz jeden rdzeń. Zarówno maszyna wirtualna łączy się komputerem głównym za pomocą kluczy SSH, jak i oprogramowanie Jenkins jest połączone z githubem za pomocą tych kluczy, a w późniejszym etapie jest to token.

2. Infrastruktura

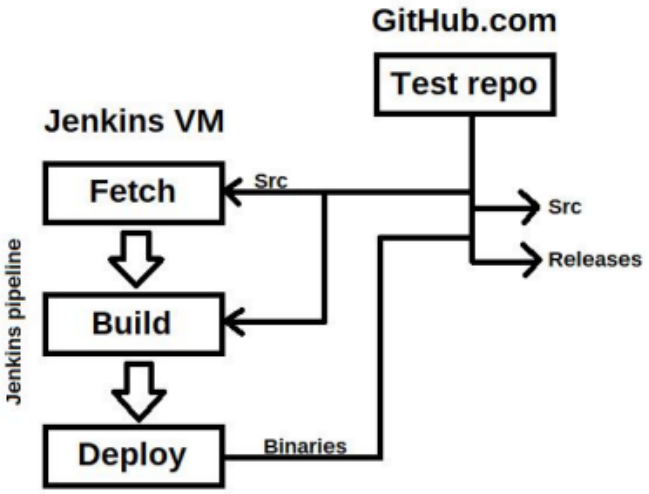


Rys 1. Infrastruktura stworzonego środowiska Jenkins-Github.

Nasza infrastruktura składa się z dwóch głównych elementów - Githuba i Jenkinsa. Github odpowiedzialny jest za przechowywanie projektu a Jenkins za jego wykonanie. Połączenie między nimi zabezpieczone jest za pomocą tokena. W

repozytorium Githuba znajduje się projekt złożony z trzech plików: Jenkinsfile, Makefile oraz hello.c. Jenkinsfile jest plikiem zawierającym listę poleceń dla Jenkinsa. Makefile zawiera pomocnicze komendy, których sam Jenkins nie jest w stanie wykonać, np. utworzenie pliku wykonywalnego z hello.c. Hello.c zawiera kod programu, który ma za zadanie wyświetlić “Hello World”.

Schemat pełnego działania umieszczono poniżej.



Rys 2. Schemat działania potoku w Jenkinsie.

3. Zdefiniowanie zagrożeń

3.1 Rozpoznane możliwe zagrożenia w naszej infrastrukturze

Ponieważ Jenkins został uruchomiony na serwerze opartym o dystrybucję Ubuntu 18.04, należało najpierw rozpatrzeć możliwe zagrożenia po stronie samej maszyny wirtualnej. Potencjalną drogą ataku na serwer jest niezabezpieczony ruch sieciowy. Możliwym atakiem jest na przykład DoS (Denial of Service), który polega na ciągłym wysyłaniu pakietów na serwer, doprowadzić ma to do wyczerpania jego zasobów, do przesycenia łącza, w konsekwencji do uniemożliwienia poprawnej pracy serwera Ubuntu wraz z usługami (tu Jenkinsem) na nim działających. Możliwym zabezpieczeniem przed atakami ze strony sieciowej jest na przykład postawienie

prawidłowo skonfigurowanego firewalla z określonymi zasadami co do ruchu wychodzącego oraz przychodzącego.

Maszyna Ubuntu została skonfigurowana jako serwer SSH. Możliwy jest do niej dostęp zdalny i sterowanie z innego hostu. Standardowym zagrożeniem w takim przypadku byłoby słabe zabezpieczenie połączenia (na przykład słabe hasło dostepu do serwera. roota). Ataki typu brute-force bazują najczęściej na prostych hasłach słownikowych, których złamanie jest proste i wymaga najczęściej tylko trochę czasu (odpowiednio ustawiony automat robi to za nas). W przypadku naszej konfiguracji, połączenie SSH zostało jednak już na początku zabezpieczone poprzez wygenerowanie pary kluczy, gdzie klucz prywatny dodatkowo zabezpieczono hasłem. W takiej sytuacji możliwym zagrożeniem jest wykradnięcie klucza prywatnego z komputera hosta. W przypadku chęci włamania na serwer jest to oczywiście zadanie dużo trudniejsze, niż samo złamanie hasła dostępu po ssh, ale mimo wszystko klucz prywatny powinien być odpowiednio zabezpieczony na komputerze, pod względem lokalizacji, a także samego hasła do klucza.

W ten sposób automatycznie poruszona jest kolejna ścieżka możliwego ataku, tj. niezabezpieczony komputer z kluczem prywatnym. W tym przypadku atak typu DoS nie jest zagrożeniem dla funkcjonowania naszego serwera Ubuntu, ale powinniśmy wprowadzić zabezpieczenia w postaci odpowiednio silnego hasła do konta, nie korzystanie domyślnie z konta administratora (z największymi uprawnieniami), zablokować ataki typu brute-force (ponieważ dzięki nim możliwe byłoby złamanie hasła dostępu do konta użytkownika), a także zainstalować na swoim komputerze dobrej jakości program anty-malware, który będzie skanował pliki w poszukiwaniu na przykład programów szpiegujących (tak zwane “keyloggery” są w stanie odczytać możliwe hasła wpisywane z klawiatury użytkownika).

Ostatnim punktem, który należy rozpatrzeć jako target potencjalnego ataku, w myśl od ogółu do szczegołu, jest sam Jenkins oraz GitHub. W przypadku obu serwisów, podstawowe zagrożenie wynika ze słabo zabezpieczonych kont użytkowników. W przypadku Jenkinsa największa podatność wynika z tego, że wystarczy prosty login i hasło, które nie zostało sformułowane zgodnie z zasadami silnego hasła, tj. hasło słownikowe, hasło zbyt krótkie, by dostęp do Jenkinsa został złamany przy pomocy brute-force. Podobnie sprawa ma się w przypadku GitHub.

Łączność między Jenkinsem a GitHubem może odbywać się poprzez klucze SSH lub wygenerowany w GitHub token. Jeśli atakujący dostanie się na naszego

Jenkinsa i/lub GitHub, może zerwać połączenie między tymi dwoma serwisami, co w sposób oczywisty uniemożliwia dalszą prawidłową pracę.

3.2 Skanowanie

Dzięki przeskanowaniu maszyny można uzyskać informacje potrzebne do ograniczenia podatności systemowych, np. otwartą liczbę portów, co pomoże zmniejszyć możliwość dostania się do maszyny przez przestępców oraz możliwości zdobycia informacji o systemie informatycznym, co skutkuje zmniejszeniem podatności, którym urządzenie może podlegać.

Nmap jest podstawowym narzędziem do skanowania portów i zabezpieczeń ale jego użycie nie przyniesie wiele informacji. Na poniższym screenie wykorzystano opcję -A, wykrywającą wersje systemu operacyjnego oraz -T4 dla szybszego działania. Adres 10.0.2.15 jest adresem Ubuntu, który został przeskanowany.

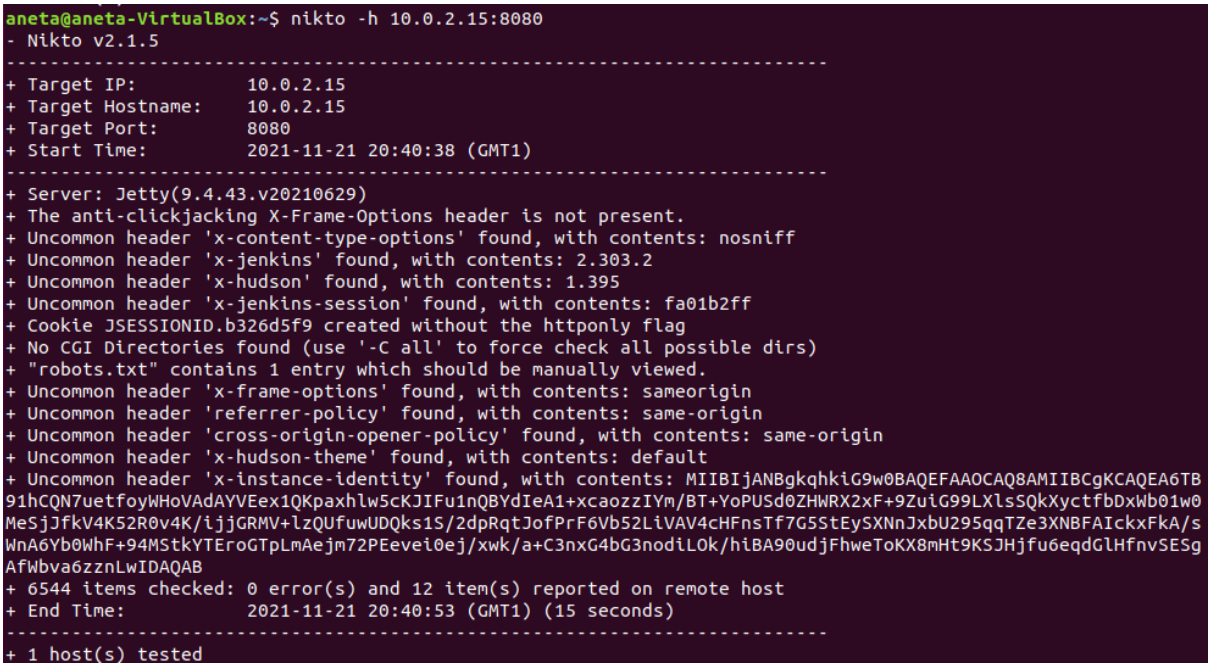
```
aneta@aneta-VirtualBox:~$ nmap -A -T4 10.0.2.15

Starting Nmap 7.60 ( https://nmap.org ) at 2021-11-21 17:22 CET
Nmap scan report for aneta-VirtualBox (10.0.2.15)
Host is up (0.000053s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 01:31:ae:a1:1d:d6:c7:a8:74:9b:03:1c:0c:f2:db:af (RSA)
|   256 87:96:7f:29:bd:d6:7d:60:34:d6:bd:c6:37:cc:40:6b (ECDSA)
|_  256 de:c4:c6:72:fc:cc:98:53:93:88:e6:01:b7:16:c4:d6 (EdDSA)
8080/tcp open  http    Jetty 9.4.43.v20210629
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: Jetty(9.4.43.v20210629)
|_http-title: Site doesn't have a title (text/html;charset=utf-8).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.62 seconds
```

Rys.3 Wynik skanu narzędzia Nmap

Nikto jest kolejnym, darmowym, skanerem podatności. Nikto wykonuje ponad 6000 testów na stronie internetowej. Potrafi znaleźć zapomniane skrypty i inne trudne do wykrycia podatności. Poniższy screen przedstawia podstawowy skan maszyny o IP: 10.0.2.15 i porcie 8080.



Rys.5 Wynik skanu Nikto

Innymi znanymi skanerami, często używanymi przy sprawdzaniu podatności, jest OpenVas oraz Nessus.

OpenVas jest darmowym narzędziem, wykorzystującym trzy lub cztery wysokie porty do aplikacji. Przy bardziej zaawansowanych skanach framework umożliwia sprawdzanie równoległe hostów, podatności oraz tworzenie własnych list portów.

Nessus jest płatnym narzędziem (z darmowym okresem próbnym), skanującym systemy operacyjne, urządzenia sieciowe, bazy danych, serwery webowe i krytyczną infrastrukturę pod kątem podatności, zagrożeń oraz zgodności. Nessus jest bardzo szybkim i dokładnym skanerem, w przejrzysty sposób opisujący wrażliwe dane systemów.

4.Wprowadzenie zabezpieczeń

W ramach praktycznej części ćwiczenia zaprezentowano poniżej przykładowe formy zabezpieczeń naszej infrastruktury.

4.1 Iptables

Iptables to narzędzie wiersza poleceń, które pozwala administratorowi systemu skonfigurować reguły filtrowania pakietów IP zapory jądra Linux. Mówiąc najprościej, iptables to program zapory ogniowej (firewall) dla systemu Linux. Monitoruje ruch z i do serwera za pomocą tabel, które zawierają zestawy reguł, zwane łańcuchami. Filtrują one przychodzące i wychodzące pakiety danych. Kiedy połączenie próbuje się ustanowić w systemie, iptables szuka reguły na liście, aby ją dopasować. Jeśli nie znajdzie żadnej to wykona domyślną akcję.

Iptables wymaga uprawnień roota do uruchomienia. W większości dystrybucji linuksowych iptables jest instalowane w katalogu /usr/sbin/iptables, jednakże w niektórych z nich można go znaleźć w /sbin/iptables.

Reguły można definiować na dwa sposoby. Pierwszym jest użycie reguł do zdefiniowania dozwolonego ruchu, a zablokowanie reszty. Drugi polega na ustawieniu reguł, aby blokowały niechciany ruch, a akceptowały pozostały. Pierwsze podejście jest częściej stosowane, ponieważ pozwala zapobiegawczo blokować nieznany, potencjalnie niebezpieczny ruch.

Polecenia dla iptables konstruujemy w następujący sposób:

iptables -t <tabela> <polecenie> <Łańcuch> [numer reguły] <specyfikacja reguły>

• -A (append) - dodaj regułę na końcu wiersza;
• -D (delete) - skasuj regułę (wymaga numeru reguły);
• -F (ush) - kasowanie wszystkich reguł z łańcucha;
• -I (insert) - wstaw regułę w odpowiednim miejscu łańcucha (wymaga numeru reguły);
• -P (policy) - ustawienie Default Policy;
• -R (replace) - zmiana specyfikacji reguły;
• -Z (zero counters) - zerowanie liczników pakietów.

Pakiet iptables operuje na tabelach:

• filter;
• mangle;

• nat.

Domyślną oraz najczęściej używaną tabelą jest filter.

Łańcuchy, które będziemy używać to: INPUT i OUTPUT.

● INPUT - Służy do kontrolowania zachowania połączeń przychodzących.

● OUTPUT - Służy do kontrolowania zachowania połączeń wychodzących.

Akcje, które będziemy używać to:

● ACCEPT - zaakceptuj (przepuść) pakiet

● DROP - zablokuj pakiet i nie wysyłaj informacji zwrotnej

● REJECT - zablokuj pakiet i wyślij informację zwrotną

● LOG - zapisz informację o pakiecie.

Poniżej umieszczono przykładowe reguły, pomagające prawidłowo zabezpieczyć maszynę przed atakami typu DoS.

Blokada komunikacji UDP w zakresie portów 500-1000 (ruch przychodzący).

Blokada komunikacji UDP zapobiega potencjalnemu atakowi UDP flood czyli atakowi typu DoS, który polega na wysyłaniu dużej liczby pakietów UDP na losowe porty.

```
iptables -A INPUT -p udp --dport 500:1000 -j LOG
--log-prefix "Blokada komunikacji UDP na portach
500:1000: "
iptables -A INPUT -p udp --dport 500:1000 -j DROP
```

Ustawienie limitu pakietów ICMP echo na max 2 na sekundę.

Ustawienie limitu pakietów ICMP echo na max 2 na sekundę uniemożliwia potencjalny atak DoS.

```
iptables -A INPUT -p icmp -m limit –limit 2/sec –
limit-burst 2 -j LOG --log-prefix "Max 2 pakiety icmp
echo na sekunde: "
```

iptables -A INPUT -p icmp -m limit –limit 2/sec –
limit-burst 2 -j ACCEPT
iptables -A INPUT -p icmp -j DROP

Bezpieczniejszym rozwiązaniem jest jednak zablokowanie całego ruchu sieciowego na maszynie Ubuntu z wyjątkiem połączenia SSH z hostem oraz połączenia z GitHubem. Ten sposób pozwala ograniczyć potencjalne wektory ataku. W takiej sytuacji należy skorzystać z przykładowych reguł iptables przedstawionych poniżej.

Dopuszczenie do połączenia SSH (na standardowym porcie) jedynie z adresem IP naszego komputera oraz do połączenia z GitHub:

```
iptables -A INPUT -p tcp --dport 22 -s adres IP komputera
-j LOG --log-prefix "Połączenie SSH przychodzące z
hostem"
iptables -A INPUT -p tcp --dport 22 -s adres IP komputera
-j ACCEPT
iptables -A OUTPUT -p tcp --dport 22 -s adres IP
komputera -j LOG --log-prefix "Połączenie SSH wychodzące
z hostem"
iptables -A OUTPUT -p tcp --dport 22 -s adres IP
komputera -j ACCEPT

iptables -t filter -A INPUT -d www.github.com -j LOG
--log-prefix "Połączenie przychodzące GitHub"
iptables -t filter -A INPUT -d www.github.com -j ACCEPT
iptables -t filter -A OUTPUT -d www.github.com -j LOG
--log-prefix "Połączenie przychodzące GitHub"
iptables -t filter -A OUTPUT -d www.github.com -j ACCEPT
```

Zablokowanie pozostałego ruchu sieciowego:

```
iptables -A INPUT -j LOG --log-prefix "Nieznane
połączenie przychodzące"
iptables -A INPUT -j DROP
iptables -A OUTPUT -j LOG --log-prefix "Nieznane
połączenie wychodzące"
iptables -A OUTPUT -j DROP
```

Zdarzenia z iptables można też rejestrować (dzięki akcjom LOG). W tym celu należy skonfigurować logowanie zdarzeń za pomocą poniższych komend.

```
sudo vi /etc/rsyslog.conf > kern.warning
/var/log/iptables.log
sudo /etc/init.d/rsyslog restart
```

Dzięki temu w pliku iptables.log można rejestrować zdarzenia w ruchu sieciowym, które będą pasować do naszych reguł. Jest to przydatne dla monitorowania stanu naszej infrastruktury (czy nie nastąpiła jakaś próba ataku).

5.Monitorowanie

5.1 Wstęp

Monitorowanie ruchu sieciowego to funkcja umożliwiająca kontrolowanie ruchu wychodzącego oraz przychodzącego. Polega ona na analizowaniu ruchu na podstawie zdefiniowanych reguł. Monitorowanie ruchu pozwala nie tylko na usprawnienie infrastruktury i naprawę błędów ale również na wykrycie niebezpiecznych zdarzeń co sprawia, że administrator sieci może szybciej zareagować na potencjalne zagrożenie.

5.2 Snort

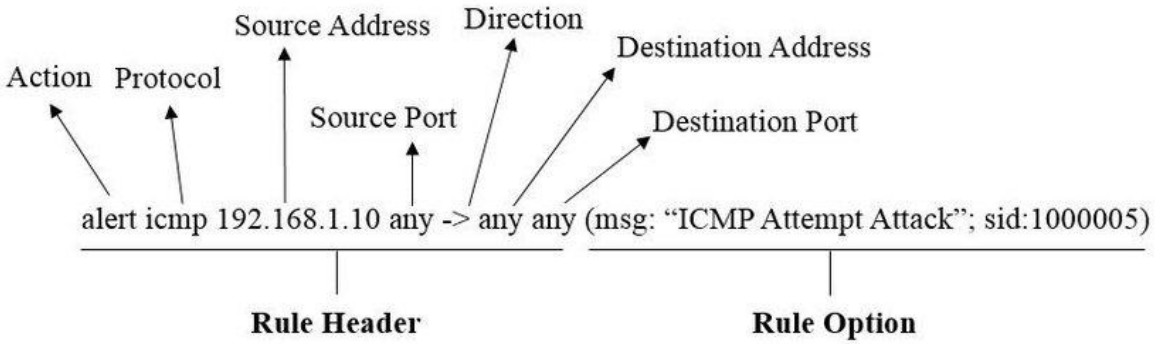
Snort jest programem typu Intrusion Detection System (IDS). Pozwalają one na wykrycie zagrożeń za pomocą odpowiednio napisanych reguł. Snort jest program typu Open Source. Pozwala to na darmowe zapoznanie się z programami IDS, tworzeniem reguł i zrozumienie ruchu sieciowego. Tworzenie reguł w Snorcie sprowadza się do tworzenia podstawowych reguł oraz bardziej rozbudowanych.

*local.rules ×

```
#alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:1000001;
rev:1; classtype:icmp-event;)
alert tcp 192.168.132.133 any -> $HOME_NET 21 (msg:"FTP connection
attempt"; sid:1000002; rev:1;)
alert tcp $HOME_NET 21 -> any any (msg:"FTP failed login";
content:"Login or password incorrect"; sid:1000003; rev:1;)
```

Rys.5 Przykładowy plik z regułami snorta

Głównymi funkcjami snorta jeśli chodzi o monitorowanie jest funkcja “alert”. Po przypisaniu odpowiednich argumentów generuje alert oraz odpowiednie logi. Bardziej okrojoną funkcją jest “log”, która jedynie generuje log. Snort posiada również takie funkcje jak pass, drop i reject. Pass ignoruje pakiet, który został wykryty. Drop odrzuca pakiet i tworzy log. Reject również odrzuca pakiet ale wysyła informację zwrotną, że pakiet został odrzucony, co nie dzieje się w przypadku “drop”.



Rys.6 Przykładowa reguła z opisem

6. Bibliografia

[1] Linux man page.

https://linux.die.net/man/8/i

[2] Encyklopedia internetowa Wikipedia - iptables.

https://pl.wikipedia.org/wiki/Iptables

[3] Wikibooks - sieci w linuksie.

https://pl.wikibooks.org/wiki/Sieci_w_Linuksie/Netfilter/iptables

[4] Poradnik do konfiguracji iptables Physd.

http://www.physd.amu.edu.pl/~m_jurga/pld/firewall

[5] Podstawowe reguły iptables IBM.

https://www.ibm.com/docs/en/linux-on-systems?topic=tests-firewall-iptables-rules