POLITECHNIKA LUBELSKA

Wydział Elektrotechniki i Informatyki

Kierunek Informatyka

PRACA MAGISTERSKA

Analiza możliwości wykorzystania interfejsu WWW do definiowania zestawu reguł zapory sieciowej

Dyplomant:

Robert Korulczyk

Nr albumu:

061323

Promotor:

dr inż. Grzegorz Kozieł

Lublin 2013

Składam serdeczne podziękowania
dr inż. Grzegorzowi Koziełowi
za poświęcony czas i pomoc
w przygotowaniu niniejszej pracy

Spis treści

1. Wstęp....5

2. Cel i zakres pracy....6

3. Opis działania zapory sieciowej w Linuksie....7

3.1. Netfilter....7

3.2. Iptables....7

3.3. Możliwości iptables....8

3.4. Zasada działania iptables....9

3.5. Składnia iptables....10

4. Technologie wykorzystane podczas tworzenia aplikacji....13

4.1. PHP....13

4.2. MySQL....15

4.3. SQLite....17

4.4. Yii....19

4.5. HTML5, CSS3 i JavaScript (jQuery)....21

5. Projekt i implementacja aplikacji....23

5.1. Powód powstania aplikacji....23

5.2. Metoda konfiguracji iptables za pomocą skryptu PHP....23

5.3. Wymagania funkcjonalne....24

5.4. Wymagania niefunkcjonalne....25

5.5. Schemat ERD....26

5.6. Struktura aplikacji....26

5.7. Hierarchia obiektów w aplikacji....27

5.8. Implementacja mechanizmu wykorzystywania zmiennych....28

6. Weryfikacja działania i testy opracowanej aplikacji....37

6.1. Instalacja....37

6.2. Wykorzystanie kont użytkowników i zestawów reguł....38

6.3. Analiza możliwości wykorzystania zmiennych globalnych....38

6.4. Analiza możliwości wykorzystania hostów 40

6.5. Analiza wykorzystania grupowania hostów....41

6.6. Zwiększenie czytelności zestawu reguł poprzez zastosowanie interfejsu WWW....42
6.7. Testy aplikacji....44
6.8. Zabezpieczenia aplikacji....46
6.9. Możliwości rozwoju aplikacji....47
7. Wnioski....48
Literatura....49

1. Wstęp

Rozwój internetu oraz jego coraz większa powszechność sprawiły, że strony internetowe przestają być jedynie źródłem informacji, a stają się coraz częściej elementem rozszerzającym funkcjonalność aplikacji komputerowych. Obecnie coraz częściej funkcje dostępne dotychczas jedynie z poziomu oprogramowania instalowanego lokalnie na komputerach użytkowników, stają się możliwe do realizacji poprzez zwykłą przeglądarkę internetową. Brak konieczności instalacji, uniezależnienie od systemu operacyjnego i dostęp z dowolnego miejsca to atuty, które przemawiają za takim rozwiązaniem.

W niniejszej pracy przedstawiony jest projekt i implementacja aplikacji wykorzystującej interfejs WWW do konfiguracji zapory w systemie Linux. Praca zawiera przegląd literatury na temat mechanizmów filtrowania pakietów w systemie Linux oraz narzędzi służących do jego konfiguracji. W dalszej części opisane zostały technologie wykorzystane do stworzenia aplikacji, która poprzez interfejs WWW pozwala na konfigurację zapory i wygenerowanie zestawu komend programu iptables.

W pracy zamieszczono opis struktury aplikacji. Przybliżone zostały zastosowane rozwiązania, a także szczegółowo wyjaśniono sposób implementacji niektórych z nich. Przeprowadzona została analiza korzyści wynikających z zastosowania interfejsu WWW do konfiguracji zapory iptables. Opisany został proces instalacji, przykładowe zastosowania oraz przetestowano działanie wygenerowanego skryptu. Testy wykazały, że zastosowany model pozwala na znacznie szybszą i wygodniejszą konfigurację zapory w systemie Linux. Dzięki zastosowanym rozwiązaniom zwiększyła się nie tylko czytelność konfiguracji, ale również dzięki dynamicznemu generowaniu reguł, poprzez wykorzystanie zmiennych, znacznie zmniejszyła się ilość wymaganej pracy.

2. Cel i zakres pracy

Celem niniejszej pracy jest zbadanie możliwości wykorzystania interfejsu WWW do definiowania reguł zapory sieciowej w systemie Linux. W tym celu zostanie stworzona aplikacja internetowa pozwalająca na graficzne przedstawienie hostów oraz reguł ich dotyczących.

Zakres pracy obejmuje następujące zagadnienia:

- Omówienie mechanizmów odpowiedzialnych za konfigurację zapory w systemie Linux;
- Omówienie technologii wykorzystanych do stworzenia graficznego interfejsu, pozwalającego na definiowanie reguł iptables;
- Stworzenie aplikacji internetowe, pozwalającej na graficzne definiowanie hostów i reguł dla zapory sieciowej;
- Testy aplikacji.

3. Opis działania zapory sieciowej w Linuksie

3.1. Netfilter

Netfilter jest frameworkiem pozwalającym na filtrowanie pakietów, translację adresów i portów sieciowych (NA(P)T) oraz manipulowanie pakietami przepływającymi przez sieć. W jądrze Linuksa znajduje się od wersji 2.4.

Netfliter jest zestawem punktów w kodzie jądra Linuksa odpowiedzialnym za obsługę sieci. Pozwala rejestrować funkcje odpowiadające za analizę i modyfikację pakietów w stosie sieciowym. Wzmiankowane punkty pogrupowane są w tablice, a funkcje przyłączone w danym punkcie nazywane są łańcuchami (tablice i łańcuchy zostaną szczegółowo opisane w dalszej części pracy). Zarejestrowana funkcja zwrotna jest wywoływana dla każdego pakietu, który przechodzi przez odpowiednie odwołania w stosie sieciowym [11].

Netfilter jest też nazwą projektu zawierającego narzędzia do konfiguracji i obsługi frameworka o tej samej nazwie. Podstawową częścią pakietu Netfilter jest iptables – aplikacja przestrzeni użytkownika, służąca do obsługi części Netfilter pracującej w warstwie IP. Inne narzędzia dostępne w ramach tego pakietu to ebtables – służące do konfiguracji Netfilter dla mostów sieciowych oraz arptables służące do zarządzania komponentami odpowiedzialnymi za filtrowanie ARP [12].

3.2. Iptables

Iptables jest konsolowym programem działającym w przestrzeni użytkownika, który pozwala na konfigurowanie reguł odpowiedzialnych za filtrowanie pakietów sieciowych. Sam w sobie nie odpowiada jednak za filtrowanie pakietów – jest jedynie narzędziem pozwalającym skonfigurować odpowiednie mechanizmy zaimplementowane w jądrze Linuksa (netfilter) [15]. Dla ułatwienia, zgodnie z przyjętą konwencją, w dalszej części pracy zapora Netfilter będzie nazywana „iptables”, ze względu na to, że właśnie to narzędzie służy do konfiguracji reguł dla ruchu sieciowego.

Narzędzie iptables zostało opracowane przez Rusty Russella w 1998 roku. Do jego stworzenia użyto języka C. Znajduje się ono w jądrze Linuksa od wersji 2.4 – wymaga jądra skompilowanego z modułem ip_tables. Program iptables może być wykorzystywany jako filtr pakietów bądź zapora stanowa kontrolująca połączenia przychodzące

i wychodzące do sieci komputerowej lub stacji roboczej [14]. Program jest skierowany do administratorów systemu (wymaga uprawnienia użytkownika root do działania) i pozwala [13]:

- wyświetlać reguły filtrowania,
- dodawać, usuwać i modyfikować reguły filtrowania,
- usuwać wszystkie reguły filtrowania z wybranej tablicy.

3.3. Możliwości iptables

Iptables pełni funkcję zapory sieciowej filtrującej ruch sieciowy i odrzucającej nieodpowiednie pakiety. Drugim zastosowaniem programu iptables jest translacja adresów sieciowych (NAT – ang. Network Address Translation) [13].

Polecenie iptables umożliwia konfigurację filtracji pakietów przy uwzględnieniu różnych parametrów, między innymi [16]:

- status połączenia: nowe, nawiązane, powiązane oraz niezwiązane z żadną sesją;
- adres MAC hosta źródłowego oraz docelowego,
- adres IP hosta źródłowego oraz docelowego,
- port TCP/UDP hosta źródłowego oraz docelowego,
- interfejs sieciowy hosta źródłowego oraz docelowego,
- zawartość pakietów w formie binarnej oraz w formie tekstowej,
- liczbę pakietów i czas połączeń,
- czas (porę) w jakiej odbywają się połączenia,
- wielkość pakietów,
- limit przesłanych danych (na przykład w MB),
- uprzednio oznaczonych przez wcześniejsze reguły,
- właściciela procesu generującego pakiety,
- fragmentację pakietu,
- wiele opcji IP i TC.

Ponadto program iptables potrafi logować wybrane pakiety, co ułatwia analizowanie reguł [16].

Filtrowanie adresów IP polega na sprawdzeniu adresu hosta źródłowego i docelowego w nagłówku pakietu IP, a następnie podjęciu odpowiednich działań w przypadku odnalezienia pakietu spełniającego zdefiniowane reguły firewalla. Filtrować

można także na podstawie protokołu – można na przykład blokować pakiety protokołu ICMP odpowiadające za działanie popularnego polecenia 'ping'. W przypadku protokołów TCP i UDP można blokować odpowiednie porty. Pozwala to zablokować użytkownikom sieci korzystanie z usług przypisanych do tych portów (na przykład WWW czy FTP).

Iptables wykorzystuje dodatkowo filtrowanie stanowe, którego zasada działania jest podobna do filtrowania pakietów, ale dodatkowo śledzi i analizuje kontekst komunikacji. Firewalle tego typu utrzymują tablice z informacjami dotyczącymi aktualnych połączeń. Filtrowanie pakietów odbywa się w warstwie sieci, natomiast filtrowanie stanowe następuje w warstwie wyższej (transportowej) modelu ISO/OSI. Dzięki temu pozwala wykryć i powstrzymać bardziej wyszukane formy ataków wykonywane za pomocą protokołów wyższego poziomu, takich jak TCP czy UDP [8].

Filtrowanie pakietów nie jest jednak tylko funkcjonalnością firewalli, pełni także ważną rolę w sterowaniu ruchem w sieci, przekazywaniem pakietów itp. [18].

3.4. Zasada działania iptables

Konfiguracja iptables polega na zdefiniowaniu szeregu reguł odpowiedzialnych za filtrowanie pakietów. Każda reguła składa się ze wzorca i akcji. Reguły pogrupowane są w tablice, które zawierają łańcuchy reguł. Tablice określają, jakim operacjom można poddać pakiety, a łańcuchy określają kierunek, w jakim te pakiety się przemieszczają. Podczas analizy pakiet jest kolejno powyrównywany ze zdefiniowanymi wzorcami reguł. Jeżeli pakiet pasuje do wzorca reguły, wykonywana jest zdefiniowana w niej akcja, jeśli zaś nie – analizowana jest zgodność pakietu z kolejnymi regułami. Działanie to będzie wykonywane iteracyjnie do momentu znalezienia odpowiedniego wzorca bądź (w przypadku braku dopasowania) wykonania domyślnej akcji [1].

Zdefiniowane w iptables tablice to [16]:

- **raw** – jest to pierwsze miejsce, do którego trafiają pakiety przed jakąkolwiek modyfikacją,
- **filter** – jest domyślną tablicą, w której dokonuje się głównej filtracji pakietów,
- **nat** – jest to tablica, w której pakiety są poddawane mechanizmowi translacji adresów,
- **mangle** – jest to tablica umożliwiająca modyfikację pakietów.

Łańcuchy zdefiniowane w iptables to [16]:

- **INPUT** – pakiety skierowane do systemu lokalnego,
- **OUTPUT** – pakiety wychodzące z systemu lokalnego,
- **FORWARD** – pakiety przekazywane do innych hostów,
- **PREROUTING** – pakiety przed routingiem,
- **POSTROUTING** – pakiety po routingu.

Podstawowe cele, jakie można określić w regułach [16]:

- **ACCEPT** – pakiet zostaje przepuszczony,
- **DROP** – pakiet zostaje odrzucony,
- **REJECT** – pakiet zostaje odrzucony, a do nadawcy zostaje wysłana informacja o rozłączeniu,
- **LOG** – pakiet zostaje zapisany w logach,
- **RETURN** – przetwarzanie bieżącego łańcucha zostaje przerwane,
- **SNAT** – zmiana adresu lub portu źródłowego,
- **DNAT** – zmiana adresu lub portu docelowego,
- **MASQURADE** – zmiana źródłowego adresu IP pakietów wychodzących na adres IP routera.

Dodatkowo, dzięki modułom rozszerzającym funkcjonalności iptables, mogą być dostępne inne cele, zdefiniowane przez te moduły. Celem może być również wcześniej zdefiniowany przez użytkownika łańcuch.

System śledzący połączenia w iptables przydziela pakietom następujące stany [16]:

- **NEW** – pakiety, które tworzą nowe połączenie,
- **ESTABLISHED** – pakiety, które należą do nawiązanego połączenia,
- **RELATED** – pakiety tworzące nowe połączenie związane z już istniejącym połączeniem,
- **INVALID** – pakiety, które nie należą do żadnego połączenia.

3.5. Składnia iptables

Jak zostało wcześniej wspomniane, iptables jest programem uruchamianym z poziomu konsoli, który służy do dodawania, edycji i usuwania reguł filtrowania pakietów. Składnię polecenia iptables możemy przedstawić następująco [1]:

```
iptables [-t tablica] komenda [wzorzec] [-j akcja]
```

Najważniejsze dostępne komendy to [17, 1]:

- `-P łańcuch polityka, --policy łańcuch polityka` – ustawienie domyślnej polityki dla łańcucha, która zostanie zastosowana, jeśli pakiet nie będzie pasował do żadnej należącej do niego reguły,
- `-A łańcuch, --append łańcuch` – dodanie reguły na koniec łańcucha,
- `-I łańcuch [nr_reguły], --insert łańcuch [nr_reguły]` – wstawienie reguły do określonego łańcucha, na określoną pozycję (jeśli pozycja nie zostanie podana, reguła zostanie wstawiona na początek podanego łańcucha),
- `-R łańcuch numer_reguły, --replace łańcuch numer_reguły` – zastąpienie reguły o podanym numerze w określonym łańcuchu inną, podaną w dalszej części polecenia,
- `-D łańcuch numer_reguły, --delete łańcuch numer_reguły` – usunięcie konkretnej reguły w określonym łańcuchu (zamiast numeru reguły można podać jej definicję),
- `-L [łańcuch], --list [łańcuch]` – wyświetlenie wszystkich reguł danego łańcucha (jeśli łańcuch nie zostanie podany, zostaną wyświetlone wszystkie reguły w wybranej tablicy),
- `-N nazwa_łańcucha, --new nazwa_łańcucha` – stworzenie nowego łańcucha o zadanej nazwie,
- `-F [łańcuch], --flush [łańcuch]` – usunięcie wszystkich reguł łańcucha (jeśli łańcuch nie zostanie podany, usunięte zostaną wszystkie reguły w wybranej tablicy),
- `-X [łańcuch], --delete-chain [łańcuch]` – kasowanie pustego łańcucha (jeśli nie zostanie podana jego nazwa, skasowane zostaną wszystkie puste łańcuchy). Skasowany może być tylko łańcuch zdefiniowany przez użytkownika, który jest pusty i do którego nie ma odwołań z innych łańcuchów,
- `-E stara_nazwa nowa_nazwa, --rename-chain stara_nazwa nowa_nazwa` – zmiana nazwy łańcucha (można zmienić nazwę tylko łańcuchów zdefiniowanych przez użytkownika).

Najważniejsze opcje specyfikujące wzorzec dopasowania to (większość opcji można negować za pomocą znaku '!') [17, 20]:

- `[!] -i interfejs, [!] --in-interface interfejs` – dopasowuje interfejs wejściowy,

- `[!] -o interfejs, [!] --out-interface interfejs` – dopasowuje interfejs wyjściowy,
- `[!] -s adres[/maska], [!] --source adres[/maska]` – dopasowuje adres źródłowy do podanego adresu hosta (możliwe jest podanie kilku adresów oddzielonych przecinkami) lub sieci w formacie CIDR,
- `[!] -d adres[/maska], [!] --destination adres[/maska]` – dopasowuje adres docelowy do podanego adresu hosta (możliwe jest podanie kilku adresów oddzielonych przecinkami) lub sieci w formacie CIDR,
- `[!] -p protokół, [!] -protocol protokół` – dopasowuje protokół warstwy transportowej (ICMP, UDP, TCP, itp.),
- `[!] --sport port[:port], [!] --source-port port[:port]` – dopasowuje numer portu źródłowego (opcja wymaga podania protokołu posiadającego porty (na przykład UDP, TCP) za pomocą opcji '-p'),
- `[!] --dport port[:port], [!] --destination-port port[:port]` – dopasowuje numer portu docelowego (opcja wymaga podania protokołu posiadającego porty (na przykład UDP, TCP) za pomocą opcji '-p'),
- `[!] -f, [!] --fragment` – dopasowuje kolejne fragmenty sfragmentowanego pakietu,
- `[!] -m moduł, [!] -match moduł` – ładuje moduł dopasowujący.

4. Technologie wykorzystane podczas tworzenia aplikacji

Przeznaczeniem tworzonej aplikacji jest generowanie reguł dla zapory systemowej w systemie Linux. W związku z tym, do jej stworzenia zostały wykorzystane technologie i oprogramowanie, które jest powszechnie dostępne i proste w instalacji na systemach z tej rodziny. Nie bez znaczenia jest też licencja, na jakiej zostało oprogramowanie wydane. System Linux, podobnie jak większość wykorzystywanego przez niego oprogramowania, rozpowszechniany jest na wolnej licencji pozwalającej użytkownikom dowolnie modyfikować i dystrybuować oprogramowanie, z którego korzystają. W poszanowaniu idei twórców Linuksa, do stworzenia aplikacji zostało wykorzystane Wolne i Otwarte Oprogramowanie, a sama aplikacja została wydana na licencji GNU GPL (GNU General Public License), która umożliwia jej dowolne modyfikowanie i dalszą dystrybucję.

4.1. PHP

PHP jest obiektowym językiem programowania zaprojektowanym głównie z myślą o generowaniu stron internetowych w czasie rzeczywistym. Jego nazwa początkowo była rozwijana jako „Personal Home Page” (Osobista Strona Domowa), zostało to jednak zmienione zgodnie z rekursywną konwencją nadawania nazw projektom GNU i obecnie oznacza „PHP Hypertext Preprocessor” (PHP Preprocesor Hipertekstu) [4]. PHP jest językiem skryptowym, co znaczy, że programy w nim napisane nie są kompilowane do kodu maszynowego zrozumiałego dla procesora, lecz są interpretowane i wykonywane przez specjalny program zwany interpreterem [22]. Choć język został stworzony do generowania stron internetowych, może być wykorzystywany do przetwarzania danych z poziomu wiersza poleceń, a także do tworzenia aplikacji działających w trybie graficznym dzięki wykorzystaniu biblioteki GTK+ i rozszerzenia PHP-GTK [23].

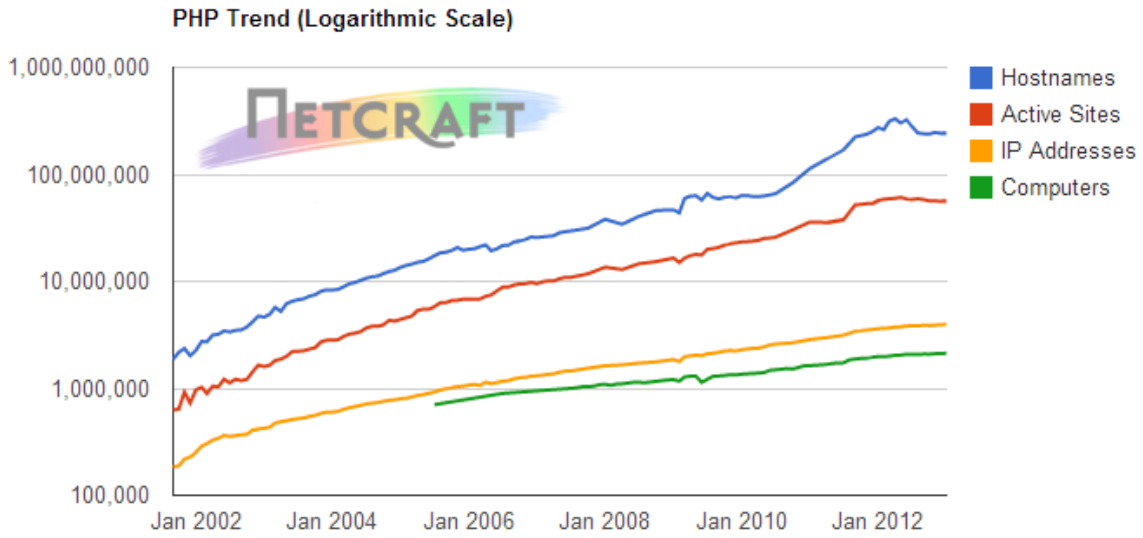
Pierwsza wersja języka została napisana w 1994 roku przez Rasmusa Lerdorfa i nosiła nazwę PHP/FI (Personal Home Page/Forms Interpreter). Początkowo projekt stworzony został w celu monitorowania odwiedzin strony autora, ale po udostępnieniu jego źródeł, język zaczął się bardzo dynamicznie rozwijać, głównie dzięki wsparciu społeczności. W 1997 roku projektem zainteresowali się dwaj izraelscy programiści: Zeev Suraski i Andi Gutmans. Postanowili całkowicie przepisać kod PHP, wykorzystując pomoc społeczności. W 1998 roku przedstawili PHP 3.0, które było następcą PHP/FI. Nowy silnik znacznie zwiększył wydajność interpretera, wprowadził elementy programowania

obiektowego, a także obsługę modułowości, dzięki czemu użytkownicy mogli rozszerzać funkcjonalność języka, poprzez dodawanie własnych modułów. Niedługo po wydaniu nowej wersji PHP 3, Zeev Suraski i Andi Gutmans postanowili ponownie przepisać kod języka. W 1999 roku przedstawili przepisany silnik interpretera języka, zwany Zend Engine. W tym samym roku powstała również założona przez nich firma Zend Technologies, która do dziś zajmuje się rozwojem i promocją języka [26]. W 2000 roku opublikowana została nowa wersja języka: PHP 4. Znacznie zwiększała ona bezpieczeństwo, a także możliwości języka udostępniając programistom wiele nowych konstrukcji językowych. W 2004 roku pojawiła się stabilna wersja PHP 5, która wprowadziła do języka wiele rozwiązań i konstrukcji dostępnych w konkurencyjnych językach, znacznie rozbudowany został również model programowania obiektowego. W kolejnych wersjach języka rozbudowana została baza modułów i rozszerzeń, wprowadzone zostały nowe elementy składni rozszerzające możliwości języka oraz ułatwiające pracę programistom. Aktualnie najnowszą wersją PHP jest 5.5.x [23].

PHP ma budowę modułową – jądro projektu zapewnia obsługę wszystkich elementów języka oraz podstawowy zestaw funkcji. Dodatkowe funkcje można dodawać instalując dodatkowe moduły [27]. Moduły można podzielić na [23]:

- **moduły jądra** – są częścią silnika PHP, są zawsze aktywne,
- **moduły oficjalne** – są elementami każdej dystrybucji PHP, aktywuje je administrator serwera,
- **moduły PECL** – są to darmowe moduły o otwartym źródle, tworzone przez społeczność i przeznaczone do samodzielnej kompilacji,
- **moduły PEAR** – zbiór klas o ujednoliconej budowie i realizujące typowe zadania.

Od pojawienia się języka, systematycznie rosła jego popularność. W 2004 roku przez PHP obsługiwane było 20% wszystkich domen sieciowych. Obecnie PHP jest najczęściej wykorzystywanym językiem do tworzenia dynamicznych stron www. Wykorzystuje go ponad 240 milionów stron internetowych [24], a jego udział wśród języków programowania wykorzystywanych do generowania stron internetowych po stronie serwera, szacuje się na ponad 80% [21].



Rysunek 4.1: Wykres przedstawiający wzrost popularności PHP (skala logarytmiczna) [Źródło: http://news.netcraft.com]

Dzięki otwartości kodu źródłowego PHP działa obecnie na większości systemów operacyjnych i pozwala na łatwą integrację z większością serwerów WWW. Dzięki wsparciu społeczności język ten bardzo dynamicznie się rozwija. Swój sukces PHP zawdzięcza również skalowalności. Prosta składnia znacznie ułatwia naukę języka początkującym programistom, a jednocześnie ogrom funkcji i gotowych bibliotek sprawia, że doświadczeni programiści znajdą wszystko, czego potrzeba do budowy zaawansowanych aplikacji. Ogromna społeczność, która zbudowała się wokół tej technologii, dostarcza wiele poradników oraz artykułów ułatwiających naukę tego języka, a mnogość dostępnych frameworków i bibliotek pozwala na tworzenie dużych i skomplikowanych systemów. Nie bez znaczenia jest też duża liczba gotowych rozwiązań udostępnianych przez twórców za darmo. Wszystko to sprawia, że PHP jest obecnie pierwszym wyborem przy tworzeniu stron internetowych i webowych systemów informatycznych [22].

4.2. MySQL

MySQL jest systemem zarządzania relacyjnymi bazami danych dostępnym na wolnej licencji. System został stworzony przez Michaela Wideniusa, rozwijany był przez szwedzką firmę MySQL AB. Jego pierwsza wersja została wydana w 1995 roku. Pomimo braku całkowitej zgodności ze standardem SQL, baza dość szybko zdobyła popularność i wkrótce stała się jednym z najczęściej wykorzystywanych systemów RDBMS (Relational Database Management System – System Zarządzania Relacyjną Bazą Danych). W 2008

roku MySQL został zakupiony przez Sun Microsystems za 1 miliard dolarów (już wtedy program był dostępny na licencji GNU GPL). W 2009 roku Oracle Corporation zostało właścicielem Sun Microsystems, a tym samym praw autorskich do MySQL i jego znaków towarowych. Posunięcie to spotkało się ze zdecydowanym sprzeciwem społeczności, powstał nawet ruch mający przeciwdziałać przejęciu MySQL-a przez Oracle. Przejęcie zostało jednak zatwierdzone, a jednym z warunków było kontynuowanie przez Oracle prac nad MySQL-em do 2015 roku [29]. Zobowiązania te nie przekonały jednak społeczności i już w 2009 roku twórca MySQL, Michael Widenius, zapoczątkował projekt o nazwie MariaDB, który był forkiem (alternatywną wersją) MySQL-a i został oparty na kodzie silnika w wersji 5.1 [28].

Obecnie równolegle do MySQL-a rozwijanego przez Oracle, rozwijany jest jego fork – MariaDB. Społeczność zbudowana wokół MySQL zarzuca Oracle, że nie dopełnia warunków umowy, kieruje rozwój projektu na zamknięte moduły i sabotuje projekt poprzez ukrywanie testów jednostkowych, co znacznie utrudnia jego rozwój przez społeczność [30]. W międzyczasie MariaDB bardzo dynamicznie się rozwija i wiele wskazuje na to, że może wkrótce zająć miejsce MySQL-a. Fedora i openSUSE (dwie popularne dystrybucje Linuksa) już zapowiedziały, że MariaDB zastąpi MySQL i będzie domyślną bazą danych w tych systemach [31]. Wkrótce można się spodziewać podobnego kroku w komercyjnych odmianach tych dystrybucji: RHEL (Red Hat Enterprise Linux) i SUSE Linux Enterprise. Bardzo prawdopodobne jest, że inne dystrybucje Linuksa pójdą ich śladem. Obecnie jednak MariaDB i MySQL są zgodne ze sobą, i projekty działające na MySQL będą działały na MariaDB [32], dlatego w poniższej pracy nie będą uwzględniane różnice pomiędzy tymi systemami. Aplikacja zostanie napisana z ukierunkowaniem na MySQL, co powinno również zagwarantować bezproblemowe działanie na MariaDB – oba systemy będzie można transparentnie wymieniać.

Najważniejsze cechy MySQL to [5]:

- relacyjność,
- architektura klient-serwer,
- niemal całkowita zgodność z ANSI SQL 99,
- obsługa podzapytań,
- obsługa widoków,
- procedury składowane,
- triggery,

- obsługa wielojęzyczności,
- wyszukiwanie pełnotekstowe,
- replikacja,
- obsługa transakcji,
- klucze obce,
- duża baza interfejsów dla języków programowania,
- niezależność od platformy – MySQL jest dostępne na najpopularniejsze systemy operacyjne,
- wydajność.

Obecnie MySQL jest najczęściej wykorzystywanim RDBMS w zastosowaniach webowych. Razem z PHP stanowią najpopularniejszą platformę do tworzenia i dystrybucji stron internetowych i systemów webowych – dostępna jest ona na niemal każdym hostingu www. Nie bez znaczenia jest także dostępność narzędzi, pozwalających na graficzne zarządzanie bazą danych. Najpopularniejsze to:

- phpMyAdmin,
- MySQL Workbench,
- Chive.

4.3. SQLite

SQLite jest systemem zarządzania relacyjnymi bazami danych zawartym w niewielkim pliku biblioteki napisanej w C. Pierwsza wersja programu została napisana przez Richarda Hippa w 2000 roku. W odróżnieniu od innych systemów RDBMS, SQLite nie działa jako osobny proces, który udostępnia interfejs dla aplikacji klienckiej, ale jest jej integralną częścią. Dzięki temu jest częstym wyborem w aplikacjach klienckich – możliwość dołączenia silnika bazy danych wraz z dostarczanym oprogramowaniem niweluje konieczność instalacji innego oprogramowania i uniezależnia go od innych aplikacji zainstalowanych w systemie, co upraszcza instalację po stronie użytkownika. Dzięki obecności API dla najpopularniejszych języków programowania, a także liberalnej licencji, implementacja takiego rozwiązania jest bardzo prosta [33]. Najpopularniejsze aplikacje wykorzystujące SQLite to [34]:

- **Adobe AIR** – multiplatformowe środowisko wykonawcze do uruchamiania aplikacji pulpitu,

- **Mozilla Firefox** – jedna z najpopularniejszych przeglądarek, wydana na wolnej licencji,
- **DBD::SQLite** – moduł Perl zgodny z DBI, dostarczający jednolity interfejs SQL,
- **Ruby on Rails** – SQLite jest domyślną bazą danych we frameworku Ruby on Rails 2.0,
- **Android** – najpopularniejszy obecnie system na smartphone'y,
- **Mac OS X** – system stworzony przez Apple,
- **Adobe Photoshop Lightroom** – program do importowania i katalogowania zdjęć z aparatów cyfrowych oraz obróbki obrazów w formacie RAW,
- **Avira Antivir** – oprogramowanie antywirusowe,
- **Avast! Antivir** – oprogramowanie antywirusowe,
- **Trac** – popularny system zarządzania projektami, wydany na licencji open source.

Zawartość bazy danych w SQLite jest przechowywana w jednym pliku – bezpieczeństwo danych jest oparte na zabezpieczeniach oferowanych przez używany system plików (choć istnieje projekt umożliwiający szyfrowanie bazy danych). SQLite implementuje większość standardu SQL 92 i udostępnia między innymi [34]:

- zapytania zagnieżdżone,
- widoki,
- klucze obce,
- transakcje,
- wyzwalacze (częściowo),
- definiowanie własnych funkcji,
- przechowywanie baz danych w pamięci RAM komputera, co znacznie przyspiesza działanie.

SQLite jest również bardzo wydajnym systemem – przy obsłudze jednego klienta potrafi wielokrotnie przewyższać inne systemy działające w architekturze klient-serwer [35], dzięki czemu doskonale sprawdza się w aplikacjach klienckich lub aplikacjach serwerowych o niewielkim obciążeniu.

W przypadku tworzonej aplikacji, SQLite zostanie wykorzystany jako domyślny silnik bazy danych, dzięki czemu zniwelowana zostanie konieczność instalacji MySQL-a oraz uprości instalację – nie będzie konieczności zakładania konta użytkownika w bazie danych dla aplikacji ani konfiguracji połączenia z bazą danych. W większości przypadków

silnik SQLite powinien zapewnić wystarczającą wydajność, dla bardziej wymagających zastosowań pozostanie możliwość wykorzystania MySQL-a jako silnika bazy danych.

4.4. Yii

Yii jest bardzo wydajnym frameworkiem PHP, przeznaczonym do szybkiego tworzenia dużych aplikacji webowych. Struktura frameworka oparta na komponentach pozwala na częstsze wykorzystywanie tego samego kodu, co znacznie przyspiesza tworzenie aplikacji i jej rozwój. Dzięki wykorzystaniu wzorców projektowych standaryzuje strukturę aplikacji, co ułatwia jej zrozumienie, wprowadzając powszechnie znane rozwiązania i standardy [36].

Twórcą frameworka jest Qiang Xue, który wcześniej tworzył inny framework PHP – PRADO. Autor nie był zadowolony z wydajności PRADO, dlatego postanowił stworzyć od podstaw nowy framework, który pozbawiony byłby niedoskonałości poprzednika. Na początku 2008 roku wydana została wersja alpha Yii, a pod koniec roku wersja stabilna oznaczona numerem 1.0. Framework czerpie pełnymi garściami z innych projektów, takich jak [37]:

- **PRADO** – jest głównym źródłem pomysłów zaimplementowanych w Yii. Yii przyjmuje po nim komponentowy i zdarzeniowy paradygmat programowania, warstwy abstrakcji bazy danych, modułową architekturę aplikacji, internacjonalizację i lokalizację oraz wiele innych jego cech i wzorów,
- **Ruby on Rails** – Yii dziedziczy po nim ducha konwencji nad konfiguracją, a także implementuje jego realizację wzorca projektowego rekordu aktywnego dla warstwy ORM,
- **jQuery** – jest to zintegrowana z Yii biblioteka JavaScript,
- **Symfony** – Yii implementuje projekt filtrów i architekturę wykorzystywania wtyczek,
- **Joomla** – Yii wykorzystuję modułową budowę i sposób tłumaczenia wiadomości.

Yii było tworzone głównie z myślą o wydajności, dzięki czemu implementuje wiele mechanizmów znacznie przyspieszających jego działanie i ułatwiających programiście optymalizację tworzonej aplikacji. Dzięki wykorzystaniu wzorca projektowego późnego ładowania znacznie ogranicza ilość ładowanych do pamięci bibliotek, co przyspiesza inicjalizację skryptów i zmniejsza ilość wykorzystywanej pamięci. Udostępniony interfejs umożliwiający buforowanie danych znacznie ułatwia optymalizację tworzonego

oprogramowania, poprzez zapisywanie i wielokrotne wykorzystywanie rezultatów elementów aplikacji generujących największe obciążenie.

Najbardziej charakterystyczne cechy Yii to [38]:

- Implementacja wzorca projektowego MVC (Model-View-Controller – Model-Widok-Kontroler);
- Implementacja WSDL do korzystania z usług sieciowych;
- Internacjonalizacja i lokalizacja (I18N oraz L10N). Yii obsługuje tłumaczenie wiadomości, dat i godzin, formatowanie liczb i lokalizację interfejsu.;
- Warstwa systemu buforowania. Obsługuje buforowanie danych, całych stron, buforowanie fragmentaryczne oraz dynamiczną zawartość w buforowanych elementach. Dzięki wykorzystaniu ujednoliconego interfejsu i dostępności wielu bibliotek implementujących różne mechanizmy buforowania, można je transparentnie zmieniać bez konieczności modyfikacji aplikacji;
- Obsługa błędów i logowania wiadomości. Błędy są przetwarzane i prezentowane w estetyczny sposób, a wiadomości dziennika mogą być kategoryzowane, filtrowane i kierowane do różnych miejsc przeznaczenia;
- Implementacja środków bezpieczeństwa, między innymi zapobieganie atakom typu cross-site scripting (XSS), cross-site request forgery (CSRF), manipulowanie ciasteczkami, itp.;
- Testy funkcjonalne i jednostkowe oparte na PHPUnit i Selenium;
- Automatyczny generator kodu dla szkieletu aplikacji oraz panelu CRUD (Create, Read, Update, Delete – Utwórz, Odczytaj, Aktualizuj, Usuń);
- Kod generowany przez komponenty Yii i narzędzia wiersza polecenia jest zgodny ze standardem XHTML;
- Starannie zaprojektowane do pracy z zewnętrznymi bibliotekami, na przykład możliwe jest użycie kodu z PEAR czy Zend Framework.

Dzięki dodatkowej warstwie abstrakcji przy kontakcie z bazą danych możliwa jest transparentna zmiana wykorzystywanego silnika bazy danych – taka możliwość zostanie wykorzystana przy implementacji dwóch silników bazy danych (MySQL i SQLite) w projektowanej aplikacji.

4.5. HTML5, CSS3 i JavaScript (jQuery)

HTML (HyperText Markup Language) jest hipertekstowym językiem znaczników powszechnie wykorzystywanym do tworzenia stron internetowych. Początki języka sięgają 1980 roku, gdzie fizyk Tim Berners-Lee, pracujący dla ośrodka badawczo-naukowego CERN stworzył prototyp hipertekstowego systemu informacyjnego służącego do wymiany dokumentów. Pierwsza publicznie dostępna specyfikacja języka HTML pojawiła się w 1991 roku. Przez lata język ewoluował, a obecnie jego rozwojem zajmuje się organizacja W3C. Choć najnowsza wersja tego języka nie ma jeszcze ukończonej specyfikacji i pozostaje na etapie ciągłego rozwoju, najpopularniejsze przeglądarki z wyprzedzeniem implementują rozwiązania, jakie wprowadza HTML5, dzięki czemu możliwe jest wykorzystywanie nowości, jakie oferuje nowy standard. Język HTML składa się z kilku kluczowych komponentów [7, 39]:

- **Znaczniki** – podstawowy komponent języka. Charakteryzują je dwie cechy: atrybuty i wartości. Wartość znacznika zawarta jest pomiędzy znacznikiem otwierającym a zamykającym, natomiast atrybuty deklarowane są wewnątrz znacznika otwierającego. Niektóre znaczniki nie posiadają wartości ani znacznika zamykającego, na przykład
;
- **Atrybuty znaczników** – są deklarowane wewnątrz znacznika otwierającego i składają się zwykle z par klucz=”wartość”.;
- **Typy danych** – przykładowe typy dostępne w HTML to: skrypty, dane arkuszy stylów, identyfikatory, nazwy, kodowania znaków, itp.;
- **Referencje znakowe i odwołania w postaci encji** – zestaw stałych pozwalających na zapisanie określonych znaków specjalnych;
- **Deklaracjie typu dokumentu** – umieszczana na początku dokumentu definiująca standard, jaki wykorzystywany jest na stronie.

Obecnie HTML jest powszechnie wykorzystywanym językiem do definiowania struktury dokumentu, natomiast do sposobu jego wyświetlania wykorzystywany jest **CSS** (Cascading Style Sheets – Kaskadowe Arkusze Stylów) – język służący do opisu formy prezentacji stron WWW. Język został opracowany przez organizację W3C w 1996 roku. Powstał w celu odseparowania warstwy prezentacji od struktury dokumentu. Arkusz stylów składa się z listy dyrektyw definiujących, w jaki sposób mają być wyświetlane dane elementy HTML lub XML. Każda reguła składa się z selektora i deklaracji stylu dla elementów, do których odnosi się selektor. Obecnie rozwijany standard CSS3 wprowadza

wiele rozwiązań znacznie zwiększających możliwości prezentacji graficznej elementów HTML (w CSS2 większość efektów można było uzyskać tylko przy użyciu zewnętrznych grafik) oraz usprawnienia obsługi urządzeń mobilnych [40].

JavaScript to skryptowy język programowania, stworzony w 1996 roku przez firmę Netscape. Głównym jego autorem jest Brendan Eich. Język najczęściej stosowany na stronach internetowych do animacji interfejsu oraz zwiększenia możliwości interakcji ze stroną poprzez dynamiczne jej modyfikowanie w reakcji na działanie użytkownika [41]. Obecnie najczęściej wykorzystywany jest poprzez jQuery – lekką bibliotekę programistyczną znacznie ułatwiającą wykorzystanie JavaScriptu, szczególnie w zakresie poruszania się po drzewie DOM i modyfikacji jego elementów. jQuery pozwala w wygodny i zrozumiały sposób korzystać z następujących funkcjonalności [42]:

- selektory – umożliwiają wybranie dowolnego podzbioru węzłów modelu DOM,
- atrybuty – jQuery pozwala przetwarzać atrybuty węzłów dokumentu,
- manipulowanie modelem DOM,
- zmiana i przypisywanie stylu do elementów,
- rozbudowana obsługa zdarzeń, możliwość definiowania własnych,
- efekty – animacje,
- AJAX – prosty interfejs realizujący asynchroniczne zapytania.

Na jQuery opartych jest wiele gotowych rozszerzeń, wykorzystywane jest też w wielu frameworkach i gotowych systemach.

Powyższe technologie zostaną wykorzystane do tworzenia interfejsu aplikacji. Pozwalają uzyskać zaawansowane efekty graficzne, przy jednoczesnym zachowaniu prostoty użytkowania. Dodatkowo, dzięki przeniesieniu na przeglądarkę logiki odpowiedzialnej za odpowiednią interpretację i wyświetlanie interfejsu, zapewniają dostępność na dowolnych urządzeniach posiadającą zgodną ze standardami przeglądarkę, bez konieczności zapewniania wsparcia dla wielu systemów operacyjnych.

5. Projekt i implementacja aplikacji

5.1. Powód powstania aplikacji

Jak zostało wcześniej wspomniane, iptables jest tekstowym programem służącym do konfiguracji filtru pakietów w systemie Linux. Tekstowy interfejs pozwala na wykonanie wszystkich niezbędnych operacji, natomiast jest mało wygodny, szczególnie w wypadku bardzo skomplikowanych zestawów reguł. O ile standardowa zapora dla pojedynczego hosta posiada z od kilku do kilkunastu reguł, to w przypadku komputerów pośredniczących w ruchu sieciowym (na przykład jako bramy sieciowe) obsługujących ruch przychodzący i wychodzący dla wielu setek, a nawet tysięcy, hostów, lista reguł potrafi rozrastać się do ogromnych rozmiarów, przez co zarządzanie nią za pomocą interfejsu tekstowego staje się bardzo niewygodne. Jednocześnie brakuje narzędzi wykorzystujących interfejs graficzny do zarządzania takimi dużymi zbiorami reguł. Powodowane jest to między innymi tym, że komputery pracujące jako bramy, bardzo często nie posiadają żadnego środowiska graficznego, co uniemożliwia korzystanie z oprogramowania wykorzystującego interfejs graficzny.

Rozwiązaniem tego problemu jest stworzenie aplikacji wykorzystującej interfejs WWW. Dzięki temu nie ma konieczności instalacji środowiska graficznego na serwerze, gdyż nie wymaga go żaden z komponentów wykorzystywanych do wygenerowania strony internetowej, działającej jako interfejs aplikacji. Samą aplikację można obsługiwać przez dowolny komputer wyposażony w graficzną przeglądarkę stron internetowych, gdyż to na nim renderowany będzie interfejs. Pozwala to więc na konfigurację zapory za pomocą zwykłego komputera ze środowiskiem graficznym, przy jednoczesnym zachowaniu całej logiki aplikacji działającej na serwerze.

5.2. Metoda konfiguracji iptables za pomocą skryptu PHP

Jednym z problemów, który pojawił się przy projektowaniu aplikacji, był sposób, w jaki za pomocą skryptu PHP skonfigurować zaporę. Konfiguracja odbywa się poprzez wydanie odpowiedniego polecenia iptables, natomiast niezbędne są do tego uprawnienia użytkownika root. Uruchamianie skryptu dostępnego przez interfejs WWW na uprawnieniach superużytkownika wiąże się ze sporym ryzykiem – potencjalna luka w skrypcie PHP może doprowadzić do przejęcia całego serwera przez osobę

wykorzystującą tę lukę. Sposób ten wymaga również zmian w standardowej konfiguracji serwera WWW, przez co nadmiernie komplikuje instalację aplikacji.

Kolejnym problemem jest sposób, w jaki zachować stworzoną konfigurację. Wprowadzone ustawienia zapory działają do momentu restartu serwera – przy ponownym uruchomieniu wszystkie reguły są zerowane i konfigurację trzeba zaczynać od początku. Iptables posiada jednak dwa polecenia, które umożliwiają eksport i import ustawień iptables. Są to [2]:

- `iptables-save` – polecenie podaje na wyjście aktualne ustawienia iptables,
- `iptables-restore` – polecenie zatwierdza podane na wejściu ustawienia iptables (wygenerowane przez polecenie `iptables-save`).

Wymusza to konieczność stworzenia skryptów, które przy zamykaniu systemu operacyjnego zapisują aktualny stan zapory do pliku, a przy jego uruchamianiu odczytują go i importują zapisane ustawienia. Rodzi to też problemy w przypadku niekontrolowanego zamknięcia systemu (na przykład przy braku zasilania) – stan zapory nie może zostać wtedy zapisany.

Rozwiązaniem tych problemów jest generowanie skryptu konfiguracyjnego zaporę. Aplikacja generuje plik z listą komend iptables, które dodają kolejne reguły. Na początku takiego skryptu znajduje się kilka komend czyszczących poprzednio zadeklarowane reguły. Plikowi takiemu wystarczy nadać uprawnienia wykonywalności, a następnie uruchomić, a zapora zostanie skonfigurowana. Taki plik może zostać dodany do skryptów startowych serwera, dzięki czemu w systemie zawsze będzie odpowiednio skonfigurowana zapora.

Podejście to też sprawia, że nie ma konieczności uruchamiania aplikacji na docelowym systemie – może ona zostać zainstalowana na dowolnym komputerze, a wygenerowane przez aplikację reguły, przeniesione na docelowy system. Ograniczamy w ten sposób listę usług na systemie docelowym, koniecznych do działania aplikacji. Uruchamianie na bramie sieciowej usługi WWW tylko po to, aby skonfigurować zaporę, jest nieuzasadnione.

5.3. Wymagania funkcjonalne

Tekstowy interfejs iptables, pomimo swojej prostoty, daje ogromne możliwości. Tworzona aplikacja nie powinna tych możliwości ograniczać. Głównym jej celem jest ułatwienie zarządzania regułami, dlatego ich poprawność nie będzie sprawdzana – przewidzenie wszystkich możliwości wykorzystania iptables jest bardzo trudne,

a nieuwzględnienie niektórych scenariuszy może uniemożliwić wykorzystania niektórych reguł.

Aplikacja musi również umożliwiać tworzenie kaskadowych reguł, które pozwalają znacznie zwiększyć wydajność zapory, Jak zostało wcześniej wspomniane, każdy pakiet jest kolejno porównywany do reguł zapory, w poszukiwaniu pasującego dopasowania. W przypadku rozbudowanej zapory może się okazać, że pakiet musi być porównywany setki razy, zanim znajdzie odpowiednią regułę. Rozwiązaniem tego jest tworzenie reguł kaskadowych, gdzie pakiet jest porównywane najpierw do reguł bardziej ogólnych, a w przypadku dopasowania do jednej z nich, do reguł szczegółowych charakterystycznych dla danej grupy. W ten sposób możliwe jest „przeskoczenie" dużej ilości reguł niepasujących do danego pakietu [1].

Głównymi funkcjonalnościami aplikacji są:

- definiowanie hostów,
- agregacja hostów w grupy, do których można przypisywać reguły,
- przypisywanie hostom dowolnie definiowanych zmiennych, które będą mogły być wykorzystywane w szablonach,
- tworzenie szablonów reguł, wykorzystujących zmienne przypisane do hostów, dzięki czemu możliwe będzie dynamiczne generowanie reguł,
- definiowanie łańcuchów.

5.4. Wymagania niefunkcjonalne

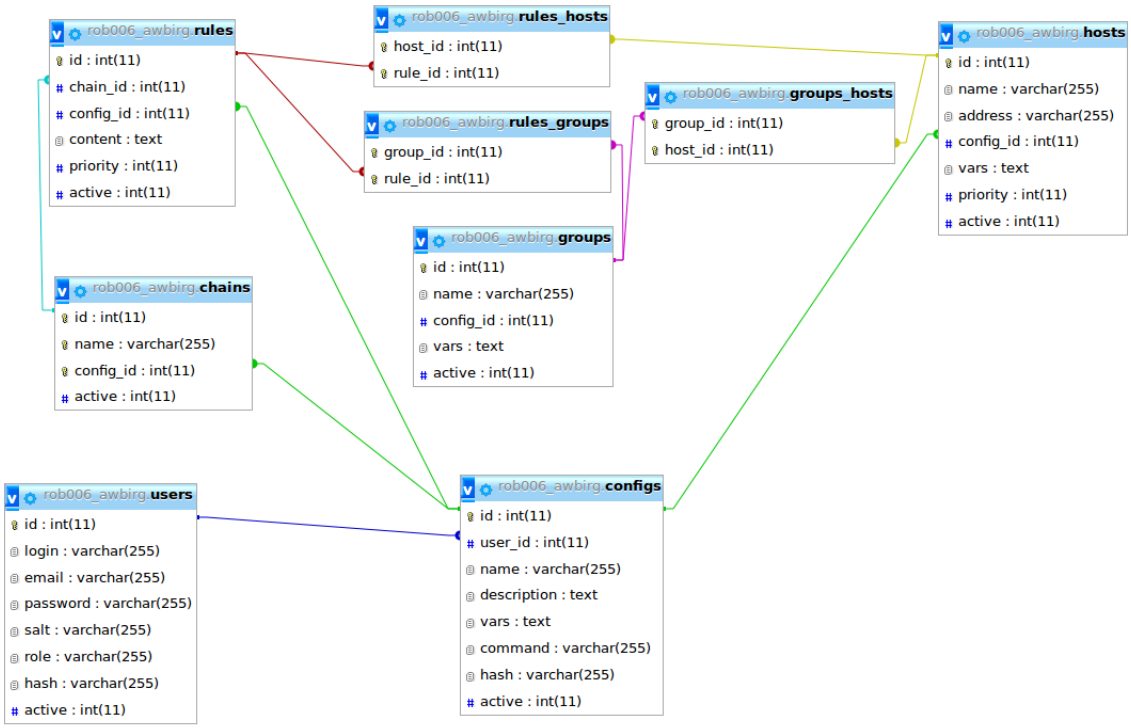
Do poprawnego zainstalowania aplikacji wymagany jest serwer z zainstalowanym:

- Apache w wersji 2.2 lub wyższej z włączoną obsługą pliku .htaccess oraz modułem mod_rewrite;
- PHP w wersji 5.2 lub wyższej;
- MySQL w wersji 5.1 lub wyższej (wymagane tylko przy wykorzystaniu MySQL jako silnika bazy danych zastępującego SQLite).

Do prawidłowego korzystania z aplikacji konieczna jest przeglądarka graficzna wspierająca HTML5, CSS3 i JavaScript, na przykład Firefox 24 oraz aktywne połączenie z serwerem, na którym zainstalowana jest aplikacja.

5.5. Schemat ERD

Na potrzeby aplikacji zaprojektowana została baza danych przechowująca informacje na temat utworzonych reguł i konfiguracji. Schemat ERD opracowanej bazy przedstawiono na rysunku 5.1.



Rysunek 5.1: Schemat ERD aplikacji

5.6. Struktura aplikacji

Aplikacja wykorzystuje wzorzec projektowy model-widok-kontroler. Poszczególne elementy aplikacji zostały podzielone ze względu na zakres odpowiedzialności na 3 grupy:

1. **Widoki** – elementy odpowiedzialne za warstwę prezentacji. Zawierają deklarację interfejsu użytkownika.
2. **Modele** – elementy odpowiedzialne za logikę aplikacji. Reprezentują dane zapisane w bazie danych, a także odpowiadają za operacje na nich dokonywane. Modele implementują również wzorzec rekordu aktywnego, co pozwala na odwoływanie się do nich tak, jakby były obiektem.
3. **Kontrolery** – elementy odpowiedzialne za przyjmowanie i obsługiwanie żądań użytkownika. Zarządzają uprawnieniami, wykonują odpowiednie operacje na modelach i wyświetlają widoki.

5.7. Hierarchia obiektów w aplikacji

W aplikacji można wyróżnić sześć podstawowych obiektów. Każdy z nich posiada panel CRUD, który pozwala na zarządzenie danymi obiektami:

1. **Użytkownicy.** Obiekty reprezentują użytkowników systemu. Wyróżnia się dwa poziomy uprawnień użytkowników: zwykły użytkownik oraz administrator. Zwykły użytkownik może tworzyć nowe i zarządzać stworzonymi przez siebie obiektami. Administrator posiada uprawnienia użytkownika, a także możliwość tworzenia i zarządzania użytkownikami istniejącymi w systemie.
2. **Konfiguracje (zestawy reguł).** Są elementami nadrzędnymi dla wymienionych niżej obiektów – aby rozpocząć pracę z aplikacją, konieczne jest stworzenie i wybranie konfiguracji. Każda konfiguracja jest przypisana do użytkownika, który ją stworzył (można przeglądać tylko swoje konfiguracje). Po wybraniu konfiguracji możliwe jest zarządzanie obiektami, które do niej należą, a także generować zestaw reguł. Obiekt posiada następujące właściwości:
    - **nazwa** – pole wymagane, pozwalające rozróżniać stworzone reguły, nie musi być jednoznaczne,
    - **komenda** – pole pozwalające określić, w jaki sposób ma być wywoływane polecenia iptables – w niektórych systemach konieczne jest podanie pełnej ścieżki do programu, domyślna wartość to „iptables”,
    - **opis** – pole pozwalające opisać tworzony zestaw,
    - **zmienne** – pole pozwalające deklarować zmienne globalne, możliwe do wykorzystania przy tworzonych regułach. Zmienne deklaruje za pomocą składni podobnej do deklarowania atrybutów w HTML: nazwa_zmiennej=”wartość_zmiennej”. W jednej zmiennej można zapisać kilka wartości, oddzielając je średnikami – dla każdej z nich zostanie wygenerowana osobna reguła wykorzystująca daną zmienną.
3. **Hosty.** Są to obiekty reprezentujące hosty istniejące w sieci. Hosty można przypisywać do reguł, dzięki czemu możliwe jest wielokrotne wykorzystywanie stworzonych szablonów. Rozwiązanie to pozwala też wyświetlać reguły przypisane dla danego hosta, co ułatwia określenie, jaka polityka jest od niego przypisana. Pola obiektu to:
    - **nazwa** – podobnie jak w przypadku konfiguracji,
    - **adres** – adres IP hosta, dostępny jako zmienna 'ip',

- **zmienne** – podobnie jak w przypadku konfiguracji, pole pozwala deklarować zmienne dostępne dla reguł przypisanych dla tego hosta.

4. **Grupy**. Obiekty grupujące hosty. Pozwalają na grupowanie i proste przypisywanie wielu hostów do reguł. Pozwala również na przypisywanie do reguł zmiennych takich samych dla każdego z hostów, na przykład listy odblokowanych portów dla grupy serwerów www. Pola obiektu to:
   - nazwa;
   - zmienne.
5. **Łańcuchy**. Obiekt grupujący reguły w odpowiednie bloki, ułatwiające tworzenie kaskadowych reguł. Do łańcucha przypisuje się reguły, z których później generowany jest łańcuch użytkownika. W ten sposób zadeklarowane łańcuchy możliwe są później do wykorzystywania przy innych regułach. Łańcuch posiada tylko jedno pole: nazwę, która musi być unikalna dla danej konfiguracji.
6. **Reguły**. Obiekt pozwalający na generowanie zestawu reguł. W regułach można stosować zmienne w postaci '{nazwa_zmiennej}' – pola te zostaną zastąpione przez wartości zmiennych dostępnych w hostach i grupach przypisanych do reguły, a także zmiennych globalnych zadeklarowanych w konfiguracji.

5.8. Implementacja mechanizmu wykorzystywania zmiennych

W aplikacji wykorzystany został autorski system zmiennych wykorzystywanych przy definiowaniu reguł. Zmienne mogą być deklarowane zarówno w konfiguracji, jak i hostach i grupach. Ponieważ deklarowane zmienne są w postaci tekstowej, przed użyciem ich przy generowaniu reguł konieczna jest ich konwersja do postaci bardziej przyjaznej, na przykład tablicy asocjacyjnej. Każdy z obiektów posiada swój własny model, który jest klasą rozszerzającą klasę `CActiveRecord`. Aby nie powielać tego samego kodu w trzech miejscach, a jednocześnie nie komplikować aplikacji kolejnymi poziomami dziedziczenia zapewniającymi daną funkcjonalność (PHP nie obsługuje wielodziedziczenia – klasa może dziedziczyć tylko po jednej klasie), wykorzystano mechanizm Zachowań (Behaviors) udostępniany przez framework Yii. Mechanizm ten pozwala na proste dodawanie metod do wybranych klas. W tym celu została stworzona klasa `GetVarsBehavior`, której kod zamieszczono na listingu 5.1.

Listing 5.1: Klasa GetVarsBehavior.

```
class GetVarsBehavior extends CActiveRecordBehavior {

  protected $_vars = null;

  public function getVars() {
    if ($this->_vars === null)
      $this->parseVars($this->getOwner()->vars);
    return $this->_vars;
  }

  protected function parseVars($model_vars) {
    $matches = array();
    preg_match_all('/([a-z_\-]+)="(.*)"/i', $model_vars, $matches,
PREG_SET_ORDER);
    $vars = array();
    foreach ($matches as $match) {
      if (empty($match[2]) || empty($match[3]))
        continue;
      $vars[$match[2]] = (strpos($match[3], ';') === false) ? $match[3] :
explode(';', $match[3]);
    }

    $this->_vars = $vars;
  }

}
```

Klasa ta rozszerza klasę `CActiveRecordBehavior`, dzięki czemu zapewnione są wszelkie mechanizmy, umożliwiające dodawanie metod do klas implementujących wzorzec rekordu aktywnego, takich jak modele wykorzystywane w aplikacji. Klasa ta posiada dwie metody:

1. `praseVars()` – prywatna metoda odpowiedzialna za parsowanie tekstu wpisanego w polu do definiowania zmiennej oraz przekształcanie znalezionych zmiennych do formatu tablicy asocjacyjnej. W tym celu wykorzystywane są wyrażenia regularne oraz wbudowana w PHP funkcja `preg_match_all()`. Metoda zwraca znalezione zmienne;
2. `getVars()` – publiczna metoda zwracająca znalezione zmienne. Wykorzystuje w tym celu metodę `parseVars()`, zwracane przez nią wyniki są zapisywane w prywatnej zmiennej `_vars`, dzięki czemu przy kolejnym jej wywołaniu nie ma konieczności powtórnego parsowania tekstu w poszukiwaniu zmiennych.

Sposób dołączenia do wybranego modelu stworzonego zachowania, został przedstawiony na listingu 5.2.

Listing 5.2: Sposób dołączenia zachowań do klasy.

```
public function behaviors() {
  return array(
    'getVars' => array(
      'class' => 'GetVarsBehavior',
    ),
  );
}
```

Dzięki takiemu zabiegowi możliwe jest wykorzystywanie metody `getVars()` z klasy `GetVarsBehavior` w każdym modelu, w którym została dodana deklaracja zaprezentowana na listingu 5.2. W ten sposób unika się powtarzalności kodu, dzięki czemu staje się łatwiejszy do zrozumienia i utrzymania. W podobny sposób można też dodawać inne funkcjonalności, wystarczy stworzone zachowania dodać do metody `behaviors()` wybranego modelu [9, 10].

Samo generowanie reguł rozpoczyna się w modelu `Config` reprezentującym daną konfigurację i odpowiada za niego kod przedstawiony na listingu 5.3.

Listing 5.3: Kod odpowiedzialny za inicjalizacje generowania reguł konfiguracji.

```
public function generateRuleSet() {
  $this->generatePreRules();
  $this->initializeChains();
  $this->generateRegularRules();
}

protected function generatePreRules() {
  foreach ($this->getPreRules() as $rule)
    $rule->addRule($this);
}

protected function initializeChains() {
  if ($this->chains) {
    foreach ($this->chains as $chain) {
      $chain->defineChain($this);
    }
    foreach($this->chains as $chain){
      $chain->generateChainRules($this);
    }
  }
}

protected function generateRegularRules() {
  foreach ($this->getRegularRules() as $rule)
    $rule->addRule($this);
}

protected function getPreRules() {
  $rules = array();
  foreach ($this->rules as $rule) {
    if ($rule->priority < 0)
      $rules[] = $rule;
  }
```

```
    return $rules;
  }

  protected function getRegularRules() {
    $rules = array();
    foreach ($this->rules as $rule) {
      if ($rule->priority >= 0)
        $rules[] = $rule;
    }
    return $rules;
  }

  public function getRuleSet() {
    return implode("\n", array_unique($this->_rule_set));
  }

  public function getLog() {
    $logs = array();
    foreach ($this->_logs as $log)
      $logs[] = $log["type"] . ': ' . $log["message"];
    return implode("\n", $logs);
  }

  public function addRule($rule) {
    if(empty($rule))
      return;

    $this->_rule_set[] = $rule;
  }

  public function addLog($message, $type = 'info') {
    if(empty($message))
      return;

    $this->_logs[] = array(
      'type' => $type,
      'message' => $message,
    );
  }
```

Generowanie reguł rozpoczyna się przy użyciu publicznej metody `generateRuleSet()`, która kolejno generuje reguły z ujemnym priorytetem (najwyższym), następnie deklaruje łańcuchy, a następnie generuje reguły z dodatnim priorytetem. Rozgraniczenie takie konieczne było ze względu na to, iż zestaw reguł często rozpoczyna się od kilku reguł czyszczących tablice ze wszystkich deklaracji, dzięki czemu możliwe jest budowanie firewalla od zera. Zadeklarowanie łańcuchów przed takimi regułami, skończyłoby się ich usunięciem, wskutek czego nie mogłyby być wykorzystywane przy budowaniu firewalla. Z kolei niemożliwe było deklarowanie łańcuchów na samym końcu, gdyż konieczne jest zadeklarowanie łańcucha przed jego wykorzystaniem, więc reguły odwołujące się do łańcuchów nie zadziałałyby. Dlatego

ujemne priorytety powinny mieć tylko reguły, które powinny zostać dodane przed deklaracją łańcuchów.

Opisany mechanizm odpowiada jedynie za wywołanie odpowiednich metod dostępnych w innych modelach, logika związana generowaniem reguł przeniesiona jest do modeli `Chain` (generowanie łańcuchów) oraz `Rule` (generowanie reguł). Kod odpowiedzialny za generowanie łańcuchów zamieszczony został na listingu 5.4.

Listing 5.4: Kod odpowiedzialny za generowanie łańcuchów użytkownika.

```
  public function defineChain(Config $config) {
    $config->addRule(Rule::generateRule($config->command, '-N ' .
$this->name));
  }

  public function generateChainRules(Config $config) {
    foreach ($this->rules as $rule)
      $rule->addRule($config, '-A ' . $this->name . " ");
  }
```

Metody przedstawione na listingu 5.4 odpowiadają kolejno za tworzenie nowego (pustego) łańcucha oraz dodawanie reguł do utworzonych łańcuchów. Takie rozgraniczenie zostało wprowadzone ze względu na możliwość odwołania się z jednego łańcucha do innego. Ponieważ nie można się odwoływać do nieistniejącego łańcucha uprzednie zadeklarowanie wszystkich łańcuchów pozwala uniknąć problemów z tym związanych. Samo dodawanie reguł wykonywane jest przez metodę `addRule()` klasy `Rule`. Kod odpowiedzialny za ten mechanizm zaprezentowano na listingu 5.5.

Listing 5.5: Kod odpowiedzialny za generowanie reguł.

```
  public function addRule(Config $config, $prefix = '', $suffix = '') {
    $this->config = $config;
    $vars = $this->findVars($this->content);
    if (empty($vars)) {
      $this->config->addRule(self::generateRule($this->config->command,
$this->content, $prefix, $suffix));
      return true;
    }

    $global_rules = $this->parseGlobalRules();
      foreach ($global_rules as $global_rule)
        $this->config->addRule(self::generateRule($this->config->command,
$global_rule, $prefix, $suffix));

    if ($this->hosts) {
      $hosts_rules = $this->parseHostsRules();
      foreach ($hosts_rules as $host_rule)
        $this->config->addRule(self::generateRule($this->config->command,
$host_rule, $prefix, $suffix));
    }
```

```
    if ($this->groups) {
      $groups_rules = $this->parseGroupsRules();
      foreach ($groups_rules as $group_rule)
        $this->config->addRule(self::generateRule($this->config->command,
$group_rule, $prefix, $suffix));
    }

    foreach ($this->log as $log)
      $this->config->addLog($log, 'notice');

    if (empty($hosts_rules) && empty($groups_rules) &&
empty($global_rules))
      $this->config->addLog('Rule cannot be generated - no data for
variables: ' . $this->content, 'warn');
  }

  protected function parseGlobalRules(){
    $vars = $this->config->getVars();
    return $this->assignVars(array($this->content), $vars, false);
  }

  protected function parseHostsRules() {
    $rules = array();
    $config_vars = $this->config->getVars();
    foreach ($this->hosts as $host) {
      $vars = $host->getVars();
      $vars['ip'] = $host->address;
      $vars = array_merge_recursive($vars, $config_vars);
      $vars = array_unique($vars, SORT_REGULAR);
      $tmp_rules = $this->assignVars(array($this->content), $vars);
      $rules = array_merge($rules, $tmp_rules);
    }
    return $rules;
  }

  protected function parseGroupsRules() {
    $rules = array();
    $config_vars = $this->config->getVars();
    foreach ($this->groups as $group) {
      $group_vars = $group->getVars();
      if (empty($group->hosts)) {
        $vars = array_merge_recursive($group_vars, $config_vars);
        $vars = array_unique($vars, SORT_REGULAR);
        $tmp_rules = $this->assignVars(array($this->content), $vars);
        $rules = array_merge($rules, $tmp_rules);
      } else {
        foreach ($group->hosts as $host) {
          $vars = $host->getVars();
          $vars['ip'] = $host->address;
          $vars = array_merge_recursive($vars, $group_vars,
$config_vars);
          $vars = array_unique($vars, SORT_REGULAR);
          $tmp_rules = $this->assignVars(array($this->content), $vars);
          $rules = array_merge($rules, $tmp_rules);
        }
      }
    }
    return $rules;
  }
```

```
  protected function assignVars($rules, &$vars, $log_errors = true) {
    $returned_rules = array();
    foreach ($rules as $rule) {
      $needed_vars = $this->findVars($rule);
      if (empty($needed_vars)) {
        $returned_rules = array_merge($returned_rules, array($rule));
      } else {
        $tmp_rules = $this->assignVar($rule, $needed_vars[0], $vars,
$log_errors);
        $tmp_rules = $this->assignVars($tmp_rules, $vars, $log_errors);
        $returned_rules = array_merge($returned_rules, $tmp_rules);
      }
    }
    return $returned_rules;
  }

  protected function assignVar($rule, $var_name, &$vars, $log_errors) {
    $rules = array();
    if (empty($vars[$var_name])) {
      if($log_errors)
        $this->log[] = 'No var {' . $var_name . '} for rule: ' . $rule;
      return $rules;
    }
    if (is_array($vars[$var_name])) {
      foreach ($vars[$var_name] as $var) {
        $rules[] = str_replace("{" . $var_name . "}", $var, $rule);
      }
    } else {
      $rules[] = str_replace("{" . $var_name . "}", $vars[$var_name],
$rule);
    }

    return $rules;
  }

  protected function findVars($content) {
    $matches = array();
    preg_match_all('/(\{[a-z_\-]+\})+/i', $content, $matches,
PREG_SET_ORDER);
    if (empty($matches))
      return false;

    $vars = array();
    foreach ($matches as $match) {
      $vars[] = str_replace(array('{', '}'), '', $match[0]);
    }
    return $vars;
  }

  public static function generateRule($command, $rule, $prefix = '',
$suffix = '') {
    return $command . " " . $prefix . $rule . $suffix . ";";
  }
```

Mechanizm przedstawiony na listingu 5.5 składa się z szeregu metod o wydzielonych zadaniach:

- `addRule()` – publiczna metoda odpowiedzialna na wygenerowanie wybranej reguły w formacie odpowiednim dla skryptu iptables. Składa się z pięciu etapów wymienionych poniżej.
    1. Jeśli metoda nie posiada żadnych zadeklarowanych zmiennych, zostaje dodana i metoda kończy swoją działanie. W przypadku gdy wykorzystywane są zmienne, wykonywane są pozostałe etapy.
    2. Parsowanie reguły z wykorzystaniem jedynie zmiennych globalnych. Pozwala to na generowane reguł wykorzystujących zmienne globalne (zadeklarowane w konfiguracji), ale nieprzypisane do żadnej grupy lub hosta. Lista takich reguł zwracana jest przez metodę `parseGlobalRules()`.
    3. Parsowanie reguł przypisanych do hostów. Wykonywana tylko dla reguł bezpośrednio przypisanych do hosta – dla każdego z nich generowane są reguły przy użyciu metody `parseHostsRules()`. Przy generowaniu takich reguł wykorzystywane są zarówno zmienne globalne, jak i te zadeklarowane w hoście przypisanym do reguły.
    4. Parsowanie reguł przypisanych do grup. Wykonywana tylko dla reguł przypisanych do jakiejś grupy – dla każdej z nich generowane są reguły przy użyciu metody `parseGroupsRules()`. Przy generowaniu takich reguł wykorzystywane są zmienne globalne, a także zmienne grupy i poszczególnych hostów należących do grupy przypisanej do reguły.
    5. Zapisywanie informacji o błędach, pozwalających na wykrycie błędów w konfiguracji, na przykład niezadeklarowanych zmiennych.
- `parseGlobalRules()` – metoda odpowiedzialna za generowanie listy reguł wykorzystujących zmienne globalne – dla każdej prawidłowej kombinacji reguły i zmiennych generowana jest jedna reguła. Przy wykorzystaniu tablic, możliwe jest więc wygenerowanie kilku reguł, dlatego metoda zwraca tablicę z listą wygenerowanych reguł. Przypisywanie zmiennych realizowane jest za pomocą metody `assignVars()`, dla której wyłączone jest rejestrowanie błędów – ponieważ krok ten jest wykonywany dla każdej reguły, te działające tylko w połączeniu z grupami lub hostami mogłyby wyświetlać mylące błędy.

- `parseHostsRules()` – metoda wykorzystywana do generowanie reguł przypisanych do hosta. Przegląda wszystkie hosty przypisane do reguły i dla każdego z nich generuje zestaw reguł z wykorzystaniem zmiennych do niego przypisanych oraz zmiennych globalnych.
- `parseGroupsRules()` – metoda wykorzystywana do generowania reguł przypisanych do grupy. Metoda przegląda grupy przypisane do reguły, a dla każdej z nich przegląda hosty do niej przypisane i generuje dla nich reguły, wykorzystując zmienne hosta, reguły oraz zmienne globalne. Metoda umożliwia również generowanie reguł dla pustej grupy – pozwala to na przykład generować reguły z wykorzystaniem zmiennych grupy bez konieczności przypisywanie ich do żądnego hosta.
- `assignVars()` – metoda przypisująca zmienne do listy reguł. Dla każdej reguły wykonuje się rekurencyjnie, aż wszystkie zmienne zostaną przypisane. Samo przypisywanie zmiennych realizowane jest przy użyciu metody `assignVar()`.
- `assignVar()` – metoda przypisuje zmienną o zadanej nazwie do zadanej reguły. W przypadku gdy zmienna jest tablicą, generuje oddzielną regułę dla każdej z nich. W przypadku gdy niemożliwe jest przypisanie zmiennej (na przykład zmienna nie istnieje) zapisuje o tym informację w logach.
- `findVars()` – metoda wyszukuje listę zmiennych użytych w regule i zwraca ich listę.
- `generateRule()` – statyczna metoda do generowania reguły w ostatecznej postaci.

6. Weryfikacja działania i testy opracowanej aplikacji

W celu sprawdzenia poprawności działania opracowanego rozwiązania przetestowano poszczególne aspekty działania aplikacji.

6.1. Instalacja

Aby zainstalować aplikację, należy skopiować pliki projektu do katalogu domeny. Do prawidłowego działania aplikacja musi posiadać prawa do zapisywania w katalogach `awbirg/protected/runtime` oraz `awbirg/assets`. Będąc w katalogu domeny można to wykonać za pomocą poleceń:

```
chmod o+w awbirg/protected/runtime/
chmod o+w awbirg/assets/
```

W przypadku gdy nie jest włączony moduł Apache `mod_rewrite`, można go włączyć wykonując następujące polecenia:

```
sudo a2enmod rewrite
sude service apache2 restart
```

Jeśli nie ma możliwości włączenia wspomnianego modułu, należy w pliku `awbirg/protected/config/main.php` zmienić linię:

```
'showScriptName' => false,
```

na:

```
'showScriptName' => true,
```

Po wykonaniu powyższych czynności aplikacja powinna działać wykorzystując bazę SQLite. W przypadku chęci korzystania z bazy MySQL konieczne jest stworzenie pliku z konfiguracją połączenia z bazą danych. Można to zrobić wykorzystując gotowy szablon poprzez polecenie:

```
cp awbirg/protected/config/base.php.template awbirg/protected/config/base.php
```

a następnie uzupełniając plik `awbirg/protected/config/base.php` danymi do połączenia z bazą danych. Po prawidłowym skonfigurowaniu połączenia należy dodać do bazy niezbędne tabele i dane początkowe. Można to zrobić wykorzystując system migracji udostępniony przez framework Yii. Całość sprowadza się do wydania polecenia:

```
php awbirg/protected/yiic.php migrate
```

a następnie potwierdzeniu operacji. Po wykonaniu tych czynności aplikacja będzie pracowała wykorzystując bazę danych MySQL. Aby powrócić do wykorzystywania SQLite należy skasować plik `awbirg/protected/config/base.php`.

Instalacja opisana na przykładzie systemu Ubuntu Serwer 12.04.

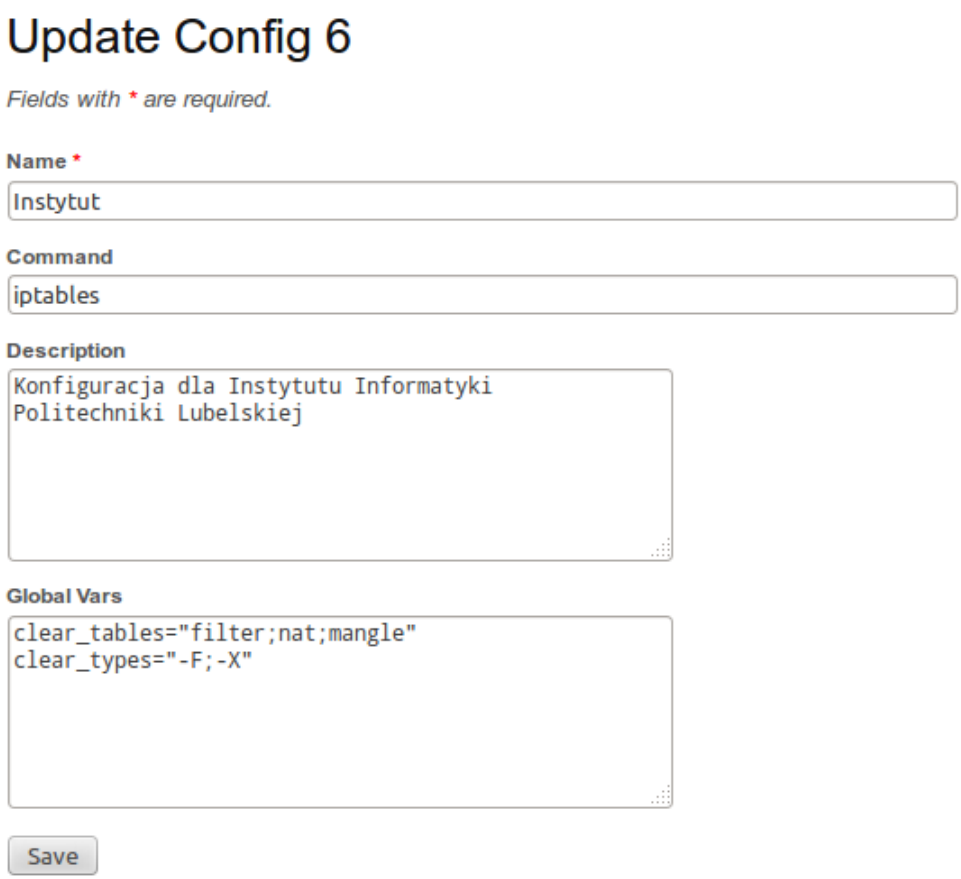
6.2. Wykorzystanie kont użytkowników i zestawów reguł

Aplikacja umożliwia tworzenie i zarządzanie kontami użytkowników. Dzięki temu możliwe jest wykorzystywanie jednej instalacji aplikacji do konfiguracji wielu sieci w sytuacji, gdy zajmują się tym inne osoby. Każdy użytkownik posiada dostęp jedynie do swoich konfiguracji, co zapobiega dostępowi niepowołanych osób do konfiguracji użytkownika. Możliwość definiowania wielu zestawów reguł przez jednego użytkownika pozwala również jednej osobie zarządzać regułami dla wielu sieci.

6.3. Analiza możliwości wykorzystania zmiennych globalnych

Zmienne globalne definiowane są w ustawieniach konfiguracji. Dostępne są dla każdej reguły, pozwalają więc definiować wartości uniwersalne. Dzięki wykorzystywaniu tablic możliwe jest definiowanie listy wartości, które mogą być wykorzystywane do generowania wielu reguł o podobnej strukturze. Najprostszym przykładem może być tutaj generowanie reguł czyszczących tablice – skrypt konfiguracyjny powinien rozpoczynać się od wyczyszczenia reguł we wszystkich istniejących tablicach. Przy trzech tablicach konieczne jest stworzenie sześciu reguł. Ponieważ jednak reguły te mają podobną strukturę, możliwe jest utworzenie jednej reguły wykorzystującej zmienne, dzięki czemu możliwe będzie wygenerowanie wymaganych reguł. Przykład takiego zastosowania widoczny jest na rysunkach 6.1 i 6.2. Rezultatem takiej konfiguracji będzie sześć reguł czyszczących:

```
iptables -t filter -F;
iptables -t filter -X;
iptables -t nat -F;
iptables -t nat -X;
iptables -t mangle -F;
iptables -t mangle -X;
```



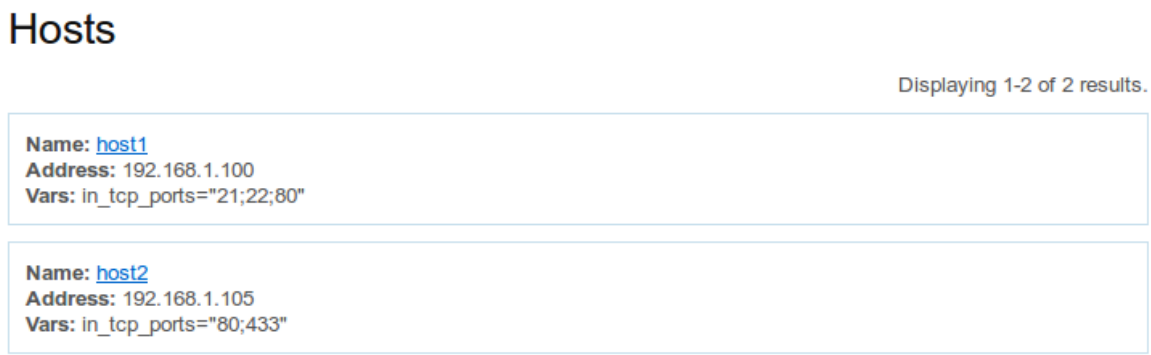
Rysunek 6.1: Widok konfiguracji ze zdefiniowanymi zmiennymi globalnymi wykorzystywanymi przy generowaniu reguł czyszczących

Rysunek 6.2: Reguła czyszcząca tablice wykorzystująca zmienne

Zmienne globalne mogą być wykorzystywane nie tylko do szybszego generowania reguł przy wykorzystaniu tablic, ale także definiowaniu wartości uniwersalnych dla każdej reguły bądź hosta, takich jak na przykład odblokowane bądź zablokowane porty. Pozwalają też definiować wartości, które mogą się w przyszłości zmienić. Przykładem może być wykorzystywany interfejs sieciowy odpowiedzialny za odbieranie i wysyłanie pakietów do sieci zewnętrznej (WAN). Jeśli interfejs zostanie podany w treści reguły, po jego zmianie konieczne będzie wprowadzenie korekt we wszystkich regułach, gdzie został wykorzystany. W przypadku wykorzystania zmiennych wystarczy zmienić definicję interfejsu jedynie w zmiennych globalnych konfiguracji.

6.4. Analiza możliwości wykorzystania hostów

Hosty są to obiekty reprezentujące poszczególne adresy IP w sieci. Dzięki możliwości zastosowania masek CIDR pozwalają również definiować grupy adresów IP, co w niektórych sytuacjach ułatwia generowanie reguł i przyspiesza ich wykonywanie – jedna reguła dla grupy adresów wykonuje się szybciej niż kilka reguł dla pojedynczych adresów IP. Podobnie jak w przypadku konfiguracji, w ramach hostów możliwe jest definiowanie zmiennych dostępnych dla reguł, do których dany host jest przypisany. Każdy host udostępnia też swój adres IP pod postacią zmiennej {ip}. W przypadku przygotowania kilku uniwersalnych reguł wykorzystujących zmienne, możliwe jest przechowywanie wszystkich charakterystycznych cech hostów pod postacią zmiennych zdefiniowanych dla każdego z nich. Znacznie zwiększa to czytelność tworzonej konfiguracji. Przykład wykorzystania tej metody można zobaczyć na rysunkach 6.3 i 6.4.



Rysunek 6.3: Lista hostów wykorzystujących zmienne do deklaracji odblokowanych portów

Rysunek 6.4: Reguła wykorzystująca zmienne do deklaracji odblokowanych portów TCP

Po połączeniu reguły z hostami zostaną wygenerowane następujące reguły:

```
iptables -A INPUT -d 192.168.1.100 -p tcp --dport 21 -j ACCEPT;
iptables -A INPUT -d 192.168.1.100 -p tcp --dport 22 -j ACCEPT;
iptables -A INPUT -d 192.168.1.100 -p tcp --dport 80 -j ACCEPT;
iptables -A INPUT -d 192.168.1.105 -p tcp --dport 80 -j ACCEPT;
iptables -A INPUT -d 192.168.1.105 -p tcp --dport 433 -j ACCEPT;
```

6.5. Analiza wykorzystania grupowania hostów

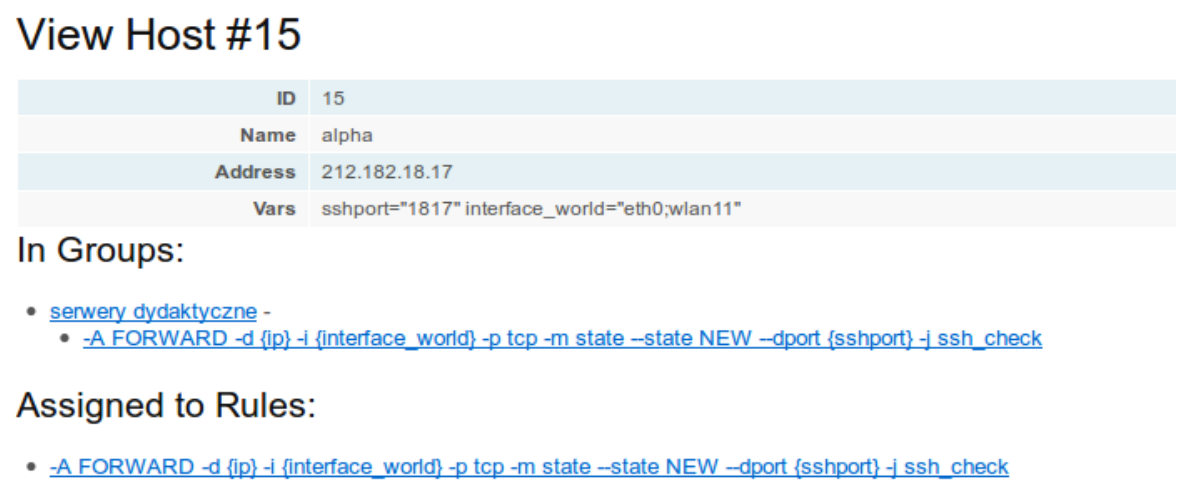
Podstawową korzyścią wynikającą z wykorzystania grupowania hostów jest ułatwienie łączenia hostów z regułami. W przypadku dużej ilości hostów tworzenie oddzielnego połączenia dla każdego hosta może być czasochłonne i pogarszać czytelność. Agregacja w grupy o podobnych właściwościach pozwala lepiej uporządkować hosty. Przykładem może być grupa hostów działających jako serwery – z racji pełnionej funkcji ich konfiguracja różnić będzie się od konfiguracji zwykłych klientów. Jednocześnie każdy serwer posiadał będzie możliwość połączenia się z nim przez ssh, co wymaga otwartego portu 22. W takim wypadku wystarczy jedynie dodać port do zmiennej zdefiniowanej w grupie. W grupie serwerów można wydzielić też kolejną grupę, która odpowiada za udostępnianie stron WWW. W analogiczny sposób można otworzyć dla niej porty 80 i 443. Dzięki takiemu podziałowi znacznie ułatwia się zrządzanie regułami, gdyż operuje się na mniejszej ilości obiektów. Jednocześnie dzięki zastosowanemu modelowi zmiennych deklarowanych w różnych obiektach, możliwe jest zarówno szybkie

przypisywanie cech dla wielu hostów, jak i definiowanie ich indywidualnie dla każdego z nich.

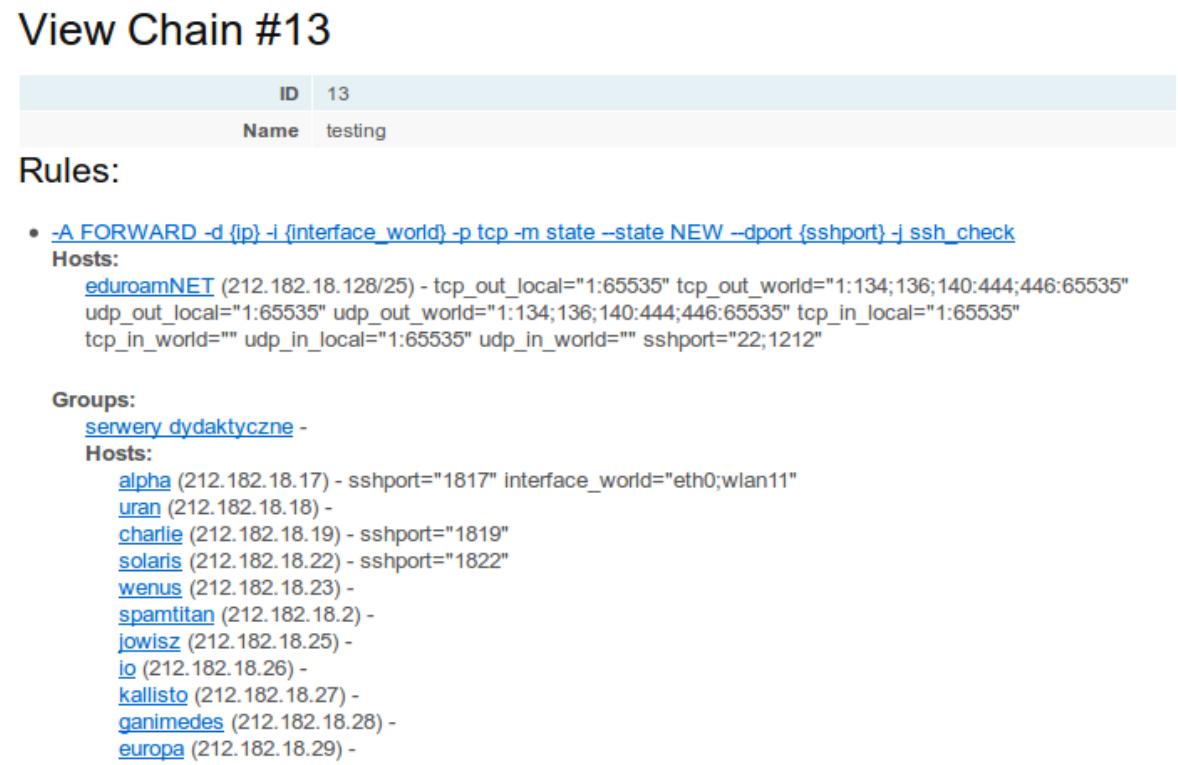
6.6. Zwiększenie czytelności zestawu reguł poprzez zastosowanie interfejsu WWW

Zastosowanie aplikacji WWW nie tylko pozwoliło na automatyzację generowania reguł, ale również zwiększyło jego czytelność. Dzięki dynamicznie generowanym widokom możliwe jest filtrowanie poszczególnych obiektów, a także wyświetlanie podsumowań dotyczących konkretnych reguł lub hostów. W widoku hosta wyświetlane są wszystkie reguły i grupy, do których jest przypisany. Pozwala to w prosty sposób określić, jak filtrowany jest ruch dotyczący tego hosta. Podobnie jest w przypadku reguł, grup oraz łańcuchów – każdy z tych obiektów posiada szczegółowy widok podsumowujący wszystkie relacje z innymi obiektami. Przykłady takich widoków można zobaczyć na rysunkach 6.5, 6.6, 6.7 i 6.8.

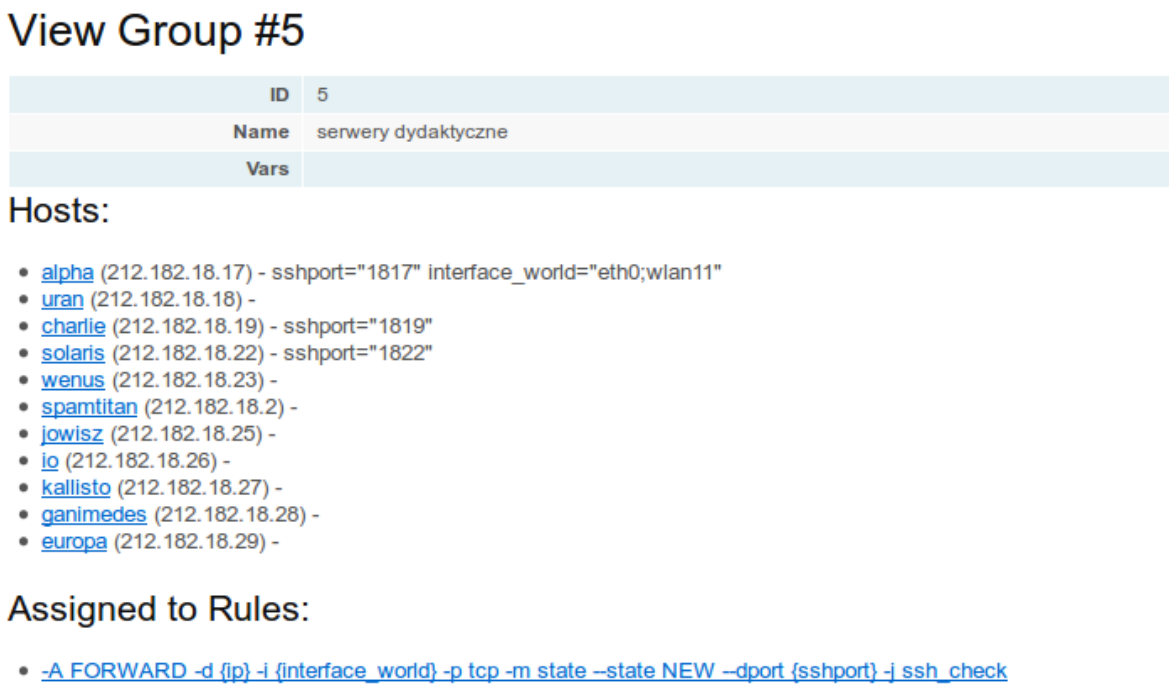
Rysunek 6.5: Widok reguły zawierający podsumowanie połączonych z nią grup i hostów



Rysunek 6.6: Widok pojedynczego hosta z podsumowaniem grup i reguł, do których jest przypisany



Rysunek 6.7: Widok łańcucha z podsumowaniem reguł do niego należących



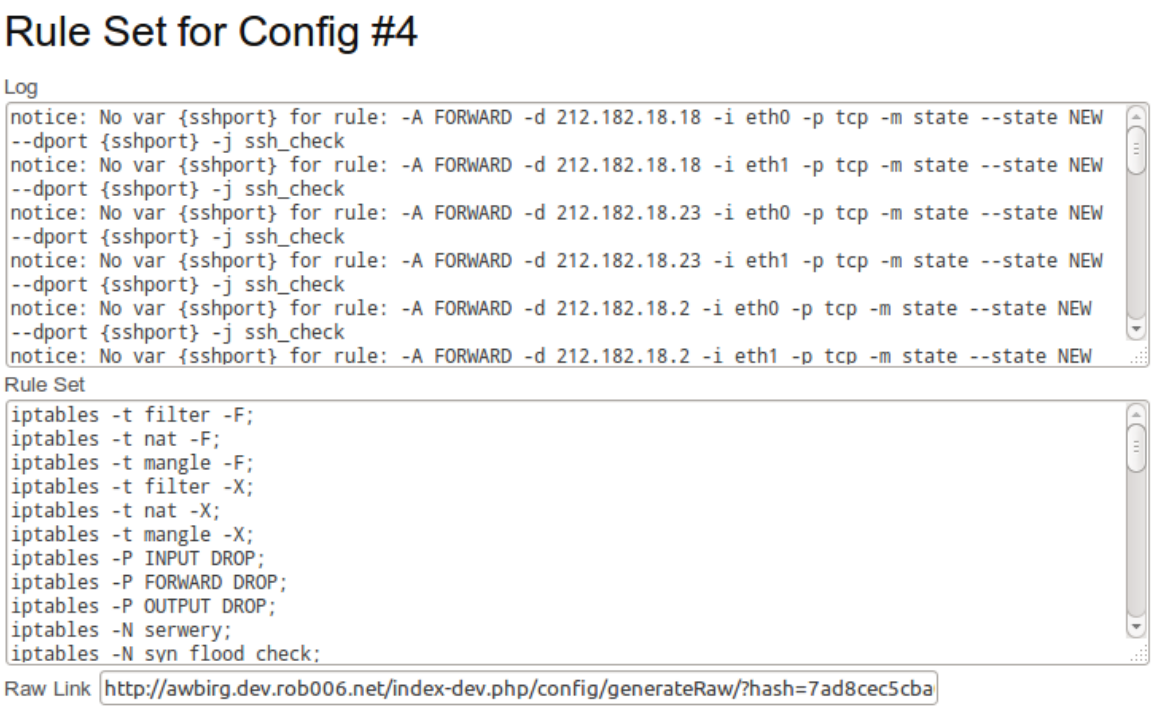
Rysunek 6.8: Widok grupy z podsumowaniem przypisanych do nie reguł i hostów

6.7. Testy aplikacji

Jak zostało wcześniej opisane, aplikacja nie sprawdza semantycznej poprawności dodawanych reguł. Reguły sprawdzane są jedynie pod kątem dostępności zmiennych przez nie wykorzystywanych. Wyróżnia się dwa rodzaje ostrzeżeń związanych z brakiem odpowiednich zmiennych:

- **notice** – jest to ostrzeżenie generowane w przypadku, gdy dla któregoś połączenia hosta z regułą nie można przypisać zmiennej;
- **warn** – jest to ostrzeżenie generowane w przypadku, gdy nie znaleziono żadnego kompletu zmiennych pozwalających wygenerować regułę.

Rezultat próby wygenerowania nieprawidłowo zadeklarowanego zestawu reguł zobaczyć można na rysunku 6.9.



Rysunek 6.9: Próba wygenerowania nieprawidłowo zadeklarowane zestawu reguł

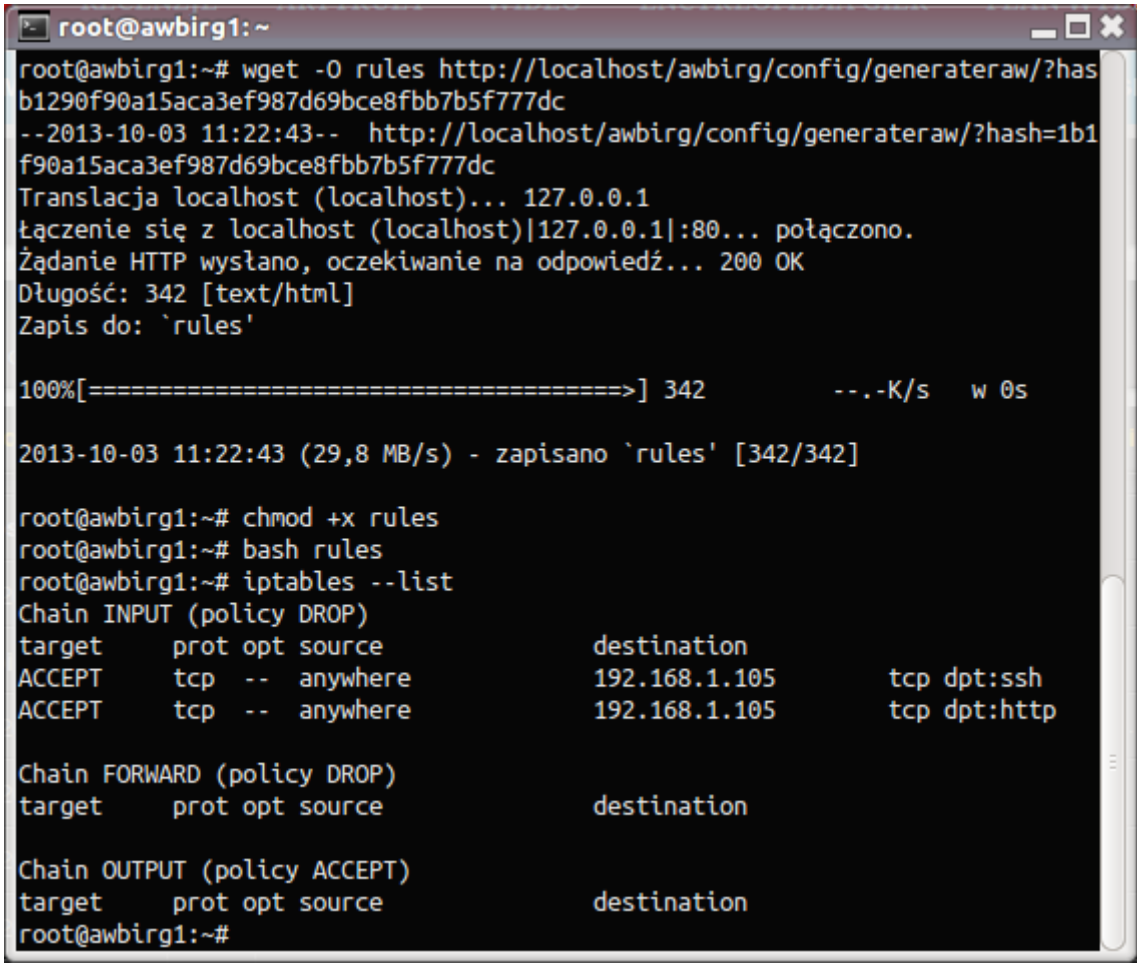
W celu przetestowania działania aplikacji stworzona została konfiguracja generująca poniższy zestaw reguł:

```
iptables -t filter -F ;
iptables -t filter -X ;
iptables -t nat -F ;
iptables -t nat -X ;
iptables -t mangle -F ;
iptables -t mangle -X ;
iptables -P INPUT DROP;
iptables -P FORWARD DROP;
iptables -P OUTPUT ACCEPT;
iptables -A INPUT -d 192.168.1.105 -p tcp --dport 22 -j ACCEPT;
iptables -A INPUT -d 192.168.1.105 -p tcp --dport 80 -j ACCEPT;
```

Zestaw pobrany został z serwera za pomocą programu „wget” i zapisany na dysku. Po nadaniu mu uprawnień do wykonywania, skrypt został uruchomiony. Skutkiem było zaimplementowanie wygenerowanych reguł, co potwierdza rezultat polecenia:

```
iptables --list
```

Przebieg procesu widoczny jest na rysunku 6.10.



Rysunek 6.10: Listing pokazujący sposób implementacji reguł i efekt ich działania

6.8. Zabezpieczenia aplikacji

Ponieważ polityka stosowana w określonej sieci może mieć kluczowe znaczenie dla jej bezpieczeństwa, ważne jest, aby zdefiniowane reguły nie wyciekły poza krąg upoważnionych osób. Aplikacja stosuje dwa sposoby uwierzytelniania:

- login oraz hasło – metoda wykorzystywana przy definiowaniu reguł. Hasło przechowywane jest w bazie danych jako hash z wykorzystaniem losowo wygenerowanej soli,
- odpowiednio przygotowany link z losowo wygenerowanym ciągiem znaków, który pozwala na pobranie listy reguł w czystej postaci za pomocą narzędzi konsolowych, takich jak na przykład `wget`.

Zastosowane rozwiązania powinny zapewnić wystarczające bezpieczeństwo. Dodatkowe środki bezpieczeństwa, jakie można zastosować to [6]:

- Zablokowanie dostępu do aplikacji z niepowołanych adresów IP. Możliwe jest to poprzez dodanie odpowiedniego wpisu w pliku .htaccess aplikacji.
- Zastosowanie szyfrowania SSL, co zapobiega atakom typu man in the middle.

6.9. Możliwości rozwoju aplikacji

Stworzona aplikacja pozwala znacznie uprościć i przyspieszyć definiowanie i zarządzanie regułami iptables. Dzięki wykorzystaniu wolnej licencji i otwartości kodu możliwe jest jej rozwijanie i dostosowywanie do własnych potrzeb. Sugerowane kierunki rozwoju to:

- Wersjonowanie konfiguracji – zapewnienie możliwości zapisywania różnych wersji danej konfiguracji, co znacznie uprości eksperymentowanie i dopracowywanie stworzonego zestawu poprzez możliwość prostego cofnięcia zmian i przywrócenia zapisanej wersji;
- Możliwość aktywacji lub dezaktywacji poszczególnych obiektów w konfiguracji, dzięki czemu możliwe będzie ich tymczasowe wyłączenie z generowanego zestawu reguł bez konieczności ich całkowitego usunięcia;
- Skrypt instalacyjny – pomimo dołożenia wszelkich starań, aby instalacja była jak najprostsza wymaga wykonania kilku skomplikowanych czynności (jak na przykład edycja plików konfiguracyjnych dla połączenia z bazą danych lub zmiana uprawnień katalogów). Czynność tą można by znacznie uprościć poprzez stworzenie prostego kreatora przeprowadzającego użytkownika przez cały proces;
- Wsparcie dla ipv6.

7. Wnioski

W pracy przedstawiono analizę możliwości wykorzystania interfejsu WWW do definiowania i generowania zestawu reguł dla zapory sieciowej w systemie Linux. Cel i zakres pracy zostały w pełni zrealizowane poprzez stworzenie aplikacji webowej umożliwiającej zarządzanie regułami iptables z poziomu przeglądarki internetowej, a następnie przetestowaniu jej działania.

Analiza wykazała, że wykorzystanie interfejsu WWW znacznie ułatwia zarządzanie dużymi zbiorami reguł dla iptables. Dzięki mechanizmowi umożliwiającemu tworzenie relacji pomiędzy konkretnymi regułami i hostami zwiększyła się czytelność stworzonego zestawu reguł. Każdy host posiada podsumowanie, pozwalające określić, jakie reguły go dotyczą, dzięki czemu znacznie prostsza stała się analiza stworzonego zestawu. Mechanizm grupowania hostów pozwala znacznie przyspieszyć tworzenie powiązań pomiędzy regułami a hostami, a także daje możliwość podziału dużej ilości hostów na grupy o podobnych cechach i właściwościach. Umożliwia również przypisywanie reguł do hostów, dzięki czemu możliwa jest ich hierarchizacja upraszczająca cały model. Dzięki możliwości definiowania łańcuchów, zapewniane jest też wsparcie dla kaskadowego modelu definiowania reguł, co pozwala na zwiększenie wydajności stworzonego zestawu.

Dużą innowacją okazał się również wykorzystywany mechanizm zmiennych. Dla każdego hosta, grupy lub konfiguracji, możliwe jest przypisywanie nazwanych wartości, które następnie będzie można wykorzystywać w przypisanych do nich regułach. Ponieważ możliwe jest deklarowanie listy wartości dostępnych pod jedną nazwą, umiejętne wykorzystanie zmiennych pozwala znacznie zaoszczędzić czas potrzebny na definiowanie reguł. Upraszcza to również cały model i czyni go bardziej intuicyjnym. Hosty, grupy oraz konfiguracje zawierają informacje, które ich bezpośrednio dotyczą, na przykład o odblokowanych lub zablokowanych portach. Reguły odpowiadają jedynie za sposób ich wykorzystania.

Stworzona aplikacja może być z powodzeniem wykorzystywana do zarządzania zbiorami reguł w dużych sieciach. Dzięki zastosowanym rozwiązaniom znacznie zwiększa czytelność, a także skraca czas potrzebny na definiowanie reguł. Wykorzystana licencja oraz otwartość kodu pozwala na dowolne jej rozwijanie i dostosowywanie do własnych potrzeb. Wykonana analiza pokazuje, że wykorzystanie interfejsu WWW do generowania zestawu reguł dla zapory iptables może przynieść znaczne korzyści.

Literatura

1. Laskowski M.: *Reorganizacja struktury sieci wewnętrznej Instytutu Informatyki Politechniki Lubelskiej w aspekcie polityki bezpieczeństwa* , Politechnika Lubelska, Lublin, 2007.
2. Negus Ch.: *Linux® Bible 2010 Edition*, Wiley Publishing Inc., Indianapolis, 2010, pages 709-713.
3. Purdy G. N.: *Linux iptables Pocket Reference*, O'Reilly Media Inc., Sebastopol, 2004.
4. Welling T.: Thomson L.: *PHP i MySQL. Tworzenie stron WWW. Vademecum Profesjonalisty*, Helion, Gliwice, 2009.
5. Kofler M.: *The Definitive Guide to MySQL 5, Third Edition*, Apress, 2005, pages 5-7.
6. Ford A.: *Apache 2 Pocket Reference: For Apache Programmers & Administrators*, O'Reilly Media Inc., Sebastopol, 2008.
7. Schafer S.M.: *HTML, XHTML i CSS. Biblia. Wydanie IV*, Helion, Gliwice, 2009.
8. Rash M.: *Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort*, No Starch Press, San Francisco, 2007.
9. Makarov A.: Yii 1.1 *Application Development Cookbook*, Packt Publishing, Birmingham, 2011.
10. Winesett J.: *Agile Web Application Development with Yii 1.1 and PHP5, Packt Publishing*, Birmingham, 2010.

Materiały uzupełniające:

11. Strona domowa projektu netfilter, http://www.netfilter.org/ [Dostęp: 09.07.2013]
12. *Netfilter*, http://pl.wikipedia.org/wiki/Netfilter [Dostęp: 09.07.2013]
13. Strona domowa projektu iptables, http://www.netfilter.org/projects/iptables/index.html [Dostęp: 09.07.2013]
14. *Iptables*, http://pl.wikipedia.org/wiki/Iptables [Dostęp: 09.07.2013]
15. *Iptables – przystępnie*, http://zsk.wsti.pl/publikacje/iptables_przystepnie.htm [Dostęp: 09.07.2013]

16. Iptables dla początkujących cz. 1, http://www.debian.pl/content/402-Iptables-dla-pocz%C4%85tkuj%C4%85cych-cz-1 [Dostęp: 10.07.2013]
17. Manual programu iptables, http://manpages.debian.net/cgi-bin/man.cgi?query=iptables [Dostęp: 12.07.2013]
18. Filtrowanie pakietów i filtrowanie stanowe, http://ljopek.kis.p.lodz.pl/ZBS/IPtables1.pdf [Dostęp: 10.07.2013]
19. Iptables dla początkujących cz. 2, http://www.debian.pl/content/403-Iptables-dla-pocz%C4%85tkuj%C4%85cych-cz-2 [Dostęp: 12.07.2013]
20. Sieci w Linuksie/Netfilter/iptables, http://pl.wikibooks.org/wiki/Sieci:Linux/Netfilter/iptables [Dostęp: 14.07.2013]
21. Usage of server-side programming languages for websites, http://w3techs.com/technologies/overview/programming_language/all [Dostęp: 17.07.2013]
22. Czym jest PHP, http://pl.wikibooks.org/wiki/PHP/Czym_jest_PHP [Dostęp: 17.07.2013]
23. PHP, http://pl.wikipedia.org/wiki/PHP [Dostęp: 17.07.2013]
24. Usage Stats for January 2013, http://php.net/usage.php [Dostęp: 17.07.2013]
25. *PHP just grows & grows*, http://news.netcraft.com/archives/2013/01/31/php-just-grows-grows.html [Dostęp: 17.07.2013]
26. *Zend Technologies*, http://en.wikipedia.org/wiki/Zend_Technologies [Dostęp: 17.07.2013]
27. *PHP – możliwości*, http://pl.wikibooks.org/wiki/PHP/Mo%C5%BCliwo%C5%9Bci [Dostęp: 18.08.2013]
28. *MySQL – historia*, http://en.wikipedia.org/wiki/MySQL#History [Dostęp: 19.07.2013]
29. *Oracle Makes Commitments to Customers, Developers and Users of MySQL*, http://www.oracle.com/us/corporate/press/042364 [Dostęp: 19.07.2013]
30. *Długi nos Larry'ego Ellisona: Pinokio, co Ty robisz z MySQL-em?*, http://www.dobreprogramy.pl/Dlugi-nos-Larryego-Ellisona-Pinokio-co-Ty-robisz-z-MySQLem,News,37784.html [Dostęp: 20.07.2013]

31. *Oracle who? Fedora & openSUSE will replace MySQL with MariaDB*, http://www.zdnet.com/oracle-who-fedora-and-opensuse-will-replace-mysql-with-mariadb-7000010640/ [Dostęp: 20.07.2013]
32. *MariaDB versus MySQL – Compatibility*, https://mariadb.com/kb/en/mariadb-vs-mysql-compatibility/ [Dostęp: 20.07.2013]
33. *SQLite*, http://en.wikipedia.org/wiki/SQLite [Dostęp: 22.07.2013]
34. *SQLite*, http://pl.wikipedia.org/wiki/SQLite [Dostęp: 22.07.2013]
35. *Database Speed Comparison*, http://sqlite.org/speed.html [Dostęp: 22.07.2013]
36. *Czym jest Yii*, http://www.yiiframework.com/doc/guide/1.1/pl/quickstart.what-is-yii [Dostęp: 24.07.2013]
37. *About Yii*, http://www.yiiframework.com/about/ [Dostęp: 24.07.2013]
38. *Yii – features*, http://en.wikipedia.org/wiki/Yii#Features [Dostęp: 24.07.2013]
39. *HTML*, http://pl.wikipedia.org/wiki/HTML [Dostęp: 25.07.2013]
40. *Kaskadowe arkusze stylów*, http://pl.wikipedia.org/wiki/Kaskadowe_arkusze_styl%C3%B3w [Dostęp: 25.07.2013]
41. *JavaScript*, http://pl.wikipedia.org/wiki/JavaScript [Dostęp: 26.07.2013]
42. *jQuery*, http://pl.wikipedia.org/wiki/JQuery [Dostęp: 26.07.2013]

Oświadczenie promotora pracy

Oświadczam, że praca została przygotowana pod moim kierownictwem naukowym i stwierdzam, że spełnia ona warunki przedstawienia jej w postępowaniu o nadanie tytułu magistra.

.....

data, podpis promotora

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej, oświadczam, że niniejsza praca została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami. Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem stopnia zawodowego/naukowego w wyższej uczelni. Niniejsza wersja pracy jest identyczna z załączoną treścią elektroniczną na CD.

.....

data, podpis autora pracy