SPECYFIKACJA WYMAGAŃ

WYMAGANIA UŻYTKOWNIKA

1. Produkt musi:
    a. realizować funkcjonalność przełącznika sieciowego lub routera IP
    b. obsługiwać dwa protokoły- MELSEC SLMP oraz Modbus TCP
    c. posiadać interfejs WWW umożliwiający: zarządzanie regułami, analizę zdarzeń, definicję reguł na poziomie rejestrów.
    d. posiadać domenę strony internetowej w protokole https
2. Produkt powinien:
    a. pracować w środowisku powszechnie używanego systemu operacyjnego
    b. zapisywać historię przepływu pakietów
2. Produkt mógłby:
    a. mógłby formatować zapisywane dane do postaci raportu

WYMAGANIA BIZNESOWE

1. Produkt musi:
    a. Stanowić dedykowane rozwiązanie problemu filtrowania ruchu protokołów przemysłowych
    b. Powstać na bazie komputera Raspberry Pi
    c. być rozbudowany o dodatkowe moduły hardware by zapewnić krytyczne funkcjonalności
2. Produkt powinien:
    a. operować na standardowych, bibliotekach Open Source
2. Produkt mógłby:
    a. być dystrybuowany Open Source

WYMAGANIA SYSTEMOWE

1. System musi:
    a. Zostać pomyślnie uruchomiony na płytce Raspberry Pi w systemie raspbian
    b. Zawierać interfejs w postaci strony https z domeną, umożliwiający wprowadzanie i wyboru reguł
    c. Posiadać implementację obsługi protokołów MELSEC SLMP i Modbus TCP
    d. Posiadać moduł filtrujący ruch protokołów przemysłowych
    e. Zawierać lokalnie uruchomiony serwer webowy
2. System powinien:
    a. Umożliwiać zapisywanie historii zapisu pakietów
4. System mógłby:
    a. Mógłby formatować dane historii zapisu pakietów do postaci raportów

Zastosowane szablony architektoniczne

Moduł CONF

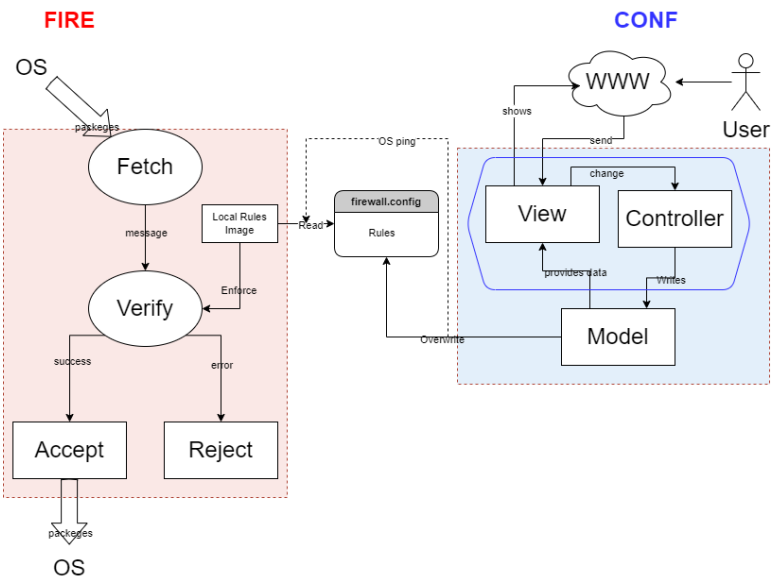
Architektura jest zrealizowana na bazie MVC zgodnie z poniższą tabelą:

	CONF
Model	Obsługuje serwer internetowy, przechwytuje komendy użytkownika.
View	Interpretuje komendę na konfigurację, przeszukuje na obecność błędów.
Controller	Pisze/Czyta zadaną konfigurację do pliku konfiguracyjnego

Moduł fire

Jest zrealizowany na bazie architektury filtrów i potoków zgodnie z poniższą tabelą:

		FIRE
INPUT		Kolejka systemowa nadchodzących pakietów
F1	**Fetch**	Ściąga pakiety z kolejki i składa w pełne wiadomości
F2	**Verify**	Weryfikuje poprawność wiadomości z regułami
OUTPUT	**Accept**	Przesyła pakiety składające się na wiadomość dalej
	Reject	Wysyła z powrotem sygnał ICMP z kodem błędu



Kluczowe elementy struktury i ich interfejsy

Moduł FIRE – Python3

Moduł FIRE odpowiada za funkcjonalną część implementowanego firewall'a. Przechwytuje on pakiety z kolejki systemowej za pomocą pakietu Netfilter. Następnie analizuje on pakiety pod kątem reguł zadanych w pliku konfiguracyjnym i podejmuje decyzję o bądź przesłaniu dalej, bądź o opuszczeniu pakietu.

Poniższa tabela przedstawia interfejs modułu FIRE:

Metoda	Zwraca	Działanie
ReadMessage()	Powodzenie – Strukturę reprezentującą pełną wiadomość, wraz z oryginalnymi pakietami na nią się składającymi Porażka – kod błędu	Czyta pakiety z kolejki systemowej, póki nie poskłada z nich wiadomości.
AnalyseMessage(mes)	TRUE -> ACCEPT FALSE -> REJECT	Analizuje wiadomość pod kątem obecnego zbioru reguł.
AcceptMessage(mes)	Powodzenie – 0 Porażka – Kod Błędu	Przepuszcza pakiety składające się na wiadomość dalej.
RejectMessage(mes)	Powodzenie – 0 Porażka – Kod Błędu	Odrzuca pakiety, wysyłając nadawcy pakiet ICMP.
UpdateConfig(dir)	Powodzenie – 0 Porażka – Kod Błędu	Aktualizuje zbiór reguł na podstawie obecnej zawartości pliku konfiguracyjnego ze ścieżki

MODUŁ CONF – PYTHON3

Moduł CONF odpowiada za możliwość konfiguracji implementowanego firewall'a. Wystawia on interfejs sieciowy umożliwiający użytkownikowi definicję nowych bądź modyfikację/usunięcie starych reguł. Po przeparsowaniu legalności działań użytkownika, moduł modyfikuje plik konfiguracyjny.

Poniższa tabela przedstawia interfejsy modułu CONF:

Metoda	Zwraca	Działanie
OPENWEB()	Powodzenie – 0 Porażka – kod błędu	Wystawia interfejs WWW
CLOSEWEB()	Powodzenie – 0 Porażka – kod błędu	Zamyka interfejs WWW
ANALYSERULE(mes)	Powodzenie – struktura Rule Porażka – Kod Błędu	Interpretuje komunikat ze strony interfejsu WWW w nową zasadę i analizuje jej legalność.
WRITERULE(RULE, DIR)	Powodzenie – 0 Porażka – Kod Błędu	Modyfikuje plik konfiguracyjny pisząc do niego legalną regułę zadaną przez użytkownika
READConfig(dir)	Powodzenie – [Rule] Porażka – Kod Błędu	Czyta z zadanego pliku konfiguracyjnego zbiór reguł i parsuje je do tablicy struktur Rule

PLIK KONFIGURACYJNY – FIREWALL.CONF

Plik konfiguracyjny odpowiada za zestaw reguł stosowanych przez moduł FIRE. Działa on w trybie White Listy, a więc zdefiniowane przez owe reguły pakiety są przepuszczane a reszta blokowana. W osobnych wierszach trzymane są definicje reguł w postaci przedstawionej w poniższej tabeli:

RuleID	**Name**	**Protocol**	**Profile**	**Direction**	**Analysed param**	**Expected Val**
SHORT	VARCHAR(50)	[MODBUS/SLPM]	SHORT	[IN/OUT/BOTH]	VARCHAR(10)	VARCHAR(50)

Powyższa reprezentacja może się zmienić podczas implementacji w zależności od wymagań struktur programowych.

Interakcje pomiędzy elementami

CONF – FIRE

Celem zastosowanej architektury była jak największa separacja modułu FIRE od CONF, aby w razie niesprawności narażonej zewnętrznie usługi sieciowej zapewnianej przez CONF, nie wyłączyć modułu FIRE odpowiadającego za bezpieczeństwo.

Mając powyższe na względzie, jedyną spodziewaną interakcją, jest wysłanie sygnału PING przez moduł CONF przy zmianie pliku konfiguracyjnego.

PLIK KONFIGURACYJNY – CONF

Plik konfiguracyjny jest czytany i pisany przez moduł CONF.

Czytanie pliku konfiguracyjnego jest na potrzeby realizacji wirtualnego środowiska zasad wewnątrz modułu, które następnie są prezentowane jak w stanie obecnym na interfejsie WWW.

Pisanie do pliku konfiguracyjnego odbywa się na żądanie autoryzowanego użytkownika, który poprzez zmianę w interfejsie WWW modyfikuje zestaw reguł. Po wykryciu takiego działania, plik jest modyfikowany by odpowiadał wymaganiom obecnym.

FIRE – PLIK KONFIGURACYJNY

Plik konfiguracyjny jest czytany przez moduł FIRE.

Po otrzymaniu ze strony systemu operacyjnego informacji o zmianie zawartości pliku konfiguracyjnego, moduł FIRE wczytuje nowy zbiór reguł. Po ich przeparsowaniu, natychmiastowo się do nich stosuje.

Określenie podstawowych mechanizmów technicznych

Sprzęt

RaspberryPi, 2x kable RJ45, Ethernet HUB.

Systemy operacyjne

Raspbian, do użytku lokalnego wystarczy linux bądź WSL.

Mechanizmy zarządzania

Udostępniony interfejs sieciowy. W wypadku działań administracyjnych, połączenie SSH do raspberry.

Mechanizmy bezpieczeństwa

Filtracja ruchu pakietów z pomocą narzędzia iptables, hasło na interfejsie sieciowym.

ANALIZA PRZYPADKÓW UŻYCIA: PRZEGLĄDANIE REGUŁ

PB1: PRZEGLĄDANIE REGUŁ STANDARDOWYCH

Aktorzy: administrator

Scenariusz główny:

1. System sprawdza tożsamość i uprawnienia użytkownika
2. System wyświetla listę zdefiniowanych przez administratora reguł:
    a. adres źródłowy
    b. adres docelowy
    c. port
    d. protokół
    e. kierunek
    f. akcja

PB2: PRZEGLĄDANIE REGUŁ ZWIĄZANYCH ZE STEROWANIEM

Aktorzy: administrator

Scenariusz główny:

1. System sprawdza tożsamość i uprawnienia użytkownika
2. System wyświetla listę zdefiniowanych przez administratora reguł:
    a. adres docelowy
    b. funkcja
    c. rejestr
    d. akcja

ANALIZA PRZYPADKÓW UŻYCIA: DODAWANIE REGUŁ

PB3: DODAWANIE REGUŁY STANDARDOWEJ

Aktorzy: administrator

Scenariusz główny:

1. System sprawdza tożsamość i uprawnienia użytkownika.
2. Administrator tworzy nową regułę z następującymi parametrami:
    a. adres źródłowy
    b. adres docelowy
    c. port
    d. protokół
    e. kierunek
    f. akcja
2. Reguła zostaje przekazana na koniec listy reguł przez system.

    Scenariusz alternatywny 1 - reguła istnieje:

    1-2. Jak w scenariuszu głównym.

3. Reguła z podanymi parametrami już istnieje, więc jej kopia nie zostanie dodana.

    Scenariusz alternatywny 2 - użytkownik nie ma uprawnień administratora:

    1-2. Jak w scenariuszu głównym.

3. Reguła nie zostaje dodana, ponieważ użytkownik ma za małe uprawnienia.

PB4: DODAWANIE REGUŁY ZWIĄZANEJ ZE STEROWANIEM

Aktorzy: administrator

Scenariusz główny:

1. System sprawdza tożsamość i uprawnienia użytkownika.
2. Administrator tworzy nową regułę z następującymi parametrami:
    a. adres docelowy
    b. funkcja
    c. rejestr
    d. akcja
2. Reguła zostaje przekazana na koniec listy reguł przez system.

Scenariusz alternatywny 1 - reguła istnieje:

1-2. Jak w scenariuszu głównym.

1. Reguła z podanymi parametrami już istnieje, więc jej kopia nie zostanie dodana.

Scenariusz alternatywny 2 - użytkownik nie ma uprawnień administratora:

1-2. Jak w scenariuszu głównym.

3. Reguła nie zostaje dodana, ponieważ użytkownik ma za małe uprawnienia.

ANALIZA PRZYPADKÓW UŻYCIA: USUWANIE REGUŁ

PB5: USUWANIE REGUŁY

Aktorzy: administrator

Scenariusz główny:

1. System sprawdza tożsamość i uprawnienia użytkownika
2. Administrator wybiera regułę, którą chce usunąć.
3. System usuwa regułę podaną regułę z listy.
4. Wszystkie reguły, które znajdowały się poniżej, zostają przesunięte o jedną pozycję w górę.

Scenariusz alternatywny - użytkownik nie ma uprawnień administratora:

1-2. Jak w scenariuszu głównym.

3. Reguła nie zostaje dodana, ponieważ użytkownik ma za małe uprawnienia.

ANALIZA PRZYPADKÓW UŻYCIA: POZOSTAŁE AKCJE

PB7: ANALIZA ZDARZEŃ BEZPIECZEŃSTWA

Aktorzy: administrator

Scenariusz główny:

1. System sprawdza tożsamość i uprawnienia użytkownika.
2. Administrator wybiera regułę, którą chce zmodyfikować i zmienia jej parametry:
    a. pozycję na liście
    b. parametry filtrowania

Scenariusz alternatywny - użytkownik nie ma uprawnień administratora:

1-2. Jak w scenariuszu głównym.

Reguła nie zostaje dodana, ponieważ użytkownik ma za małe uprawnienia.

FU1: LOGOWANIE UŻYTKOWNIKA

Aktorzy: użytkownik, serwer WWW

Scenariusz główny:

1. Użytkownikowi zostaje wyświetlony formularz do zalogowania się do systemu.
2. Użytkownik wypełnia formularz i przesyła do weryfikacji.
3. Użytkownik uzyskuje dostęp do panelu administratora.

Scenariusz poboczny - użytkownik nie podał poprawnych danych:

1-2. Jak w scenariuszu głównym.

3. Użytkownik nie uzyskuje dostępu do panelu administratora.

Scenariusz poboczny - użytkownik nie jest administratorem.

1-2. Jak w scenariuszu głównym.

3. Użytkownik nie uzyskuje pełnego dostępu do panelu administratora - może jedynie przeglądać reguły.

FU2: WYŚWIETLENIE AKTUALNEJ LISTY REGUŁ NA SERWERZE WWW

Aktorzy: administrator, serwer WWW

Scenariusz główny:

1. Administrator otwiera okno reguł na serwerze.
2. System pobiera listę reguł zapisanych w pliku konfiguracyjnym.
3. System wyświetla listę reguł administratorowi.

FU3: MODYFIKOWANIE LISTY REGUŁ NA SERWERZE WWW

Aktorzy: administrator, serwer WWW, firewall

Scenariusz główny:

1-3. Jak w FU2.

1. Administrator wybiera rodzaj reguły.
2. Administrator wprowadza modyfikacje listy reguł.
3. Zaktualizowana lista reguł zostaje zapisana w pliku.
4. Firewall zostaje poinformowany o zmianie przez serwer WWW.
5. Firewall ładuje zmiany.

FU4: ANALIZA ZAPYTAŃ Z SERWERA PRZEZ FIREWALL

Aktorzy: firewall, komputer sterujący

Scenariusz główny:

1. Interfejs sieciowy firewalla przyjmuje pakiety sieciowe i umieszcza je w kolejce wejściowej.
2. Pakiety z kolejki wejściowej przekazywane są do analizy nagłówków pod kątem zgodności z regułami filtracji sieciowej.
3. Odebrane z serwera zapytanie następnie analizowane jest pod kątem zgodności z regułami filtracji zapytań do sterowania.
4. Przefiltrowane zapytanie jest wysyłane do komputera sterującego.

Scenariusz poboczny 1 - pakiet jest odrzucony na poziomie filtracji pakietów:

1-3. Jak w scenariuszu głównym

4. Pakiety zostają odrzucone.
5. Szczegóły zapisywane są do logów.

6. Do hosta źródłowego odsyłana jest informacja o odrzuceniu.

Scenariusz poboczny 2 - pakiet jest odrzucony na poziomie analizy zapytań:

1-4. Jak w scenariuszu głównym

5. Zapytanie zostają odrzucone.
6. Szczegóły zapisywane są do logów.
7. Do hosta źródłowego odsyłana jest informacja o odrzuceniu.

FU5: ANALIZA ZDARZEŃ

Aktorzy: administrator, serwer WWW

Scenariusz główny:

1. Administrator wyznacza okres analizy zdarzeń.
2. System pobiera logi z wyznaczonego okresu.
3. Na podstawie listy reguł i logów tworzony jest widok historii zdarzeń.
4. Historia zdarzeń wyświetlona jest administratorowi.

Scenariusz poboczny 1 - pakiet jest odrzucony na poziomie filtracji pakietów

1-3. Jak w scenariuszu głównym

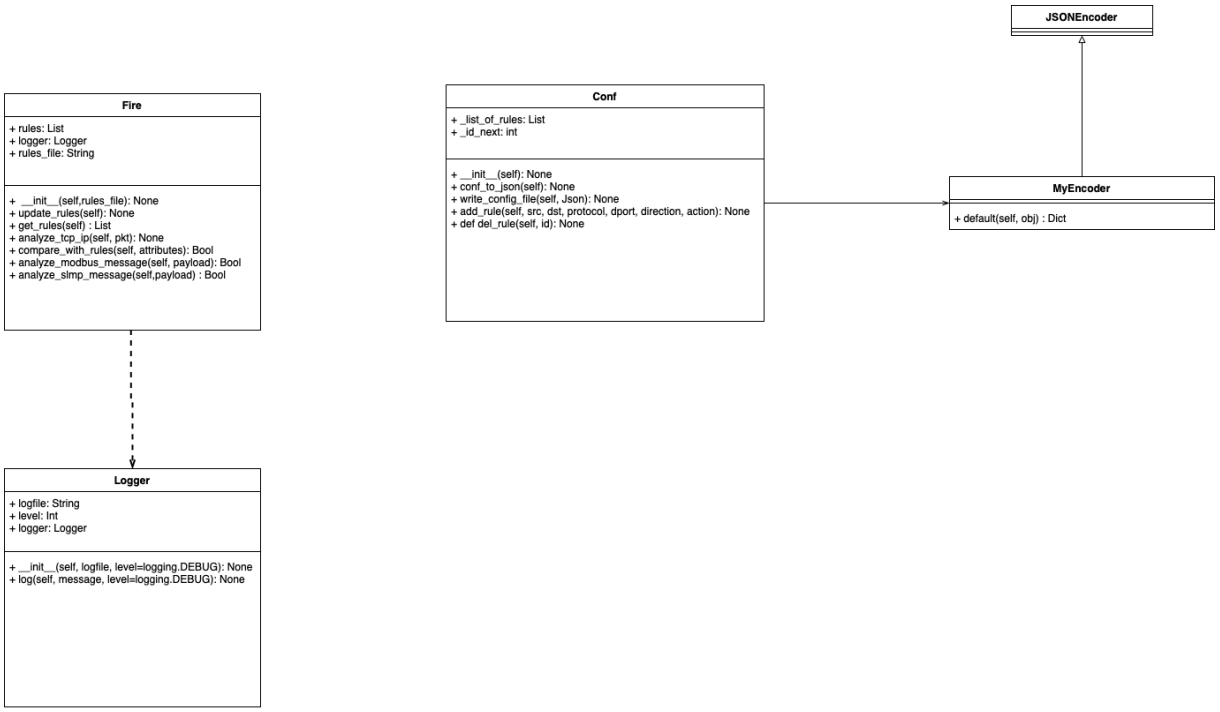
1. Pakiety zostają odrzucone.
2. Szczegóły zapisywane są do logów.
3. Do hosta źródłowego odsyłana jest informacja o odrzuceniu.

METODY REALIZACJI PROJEKTU

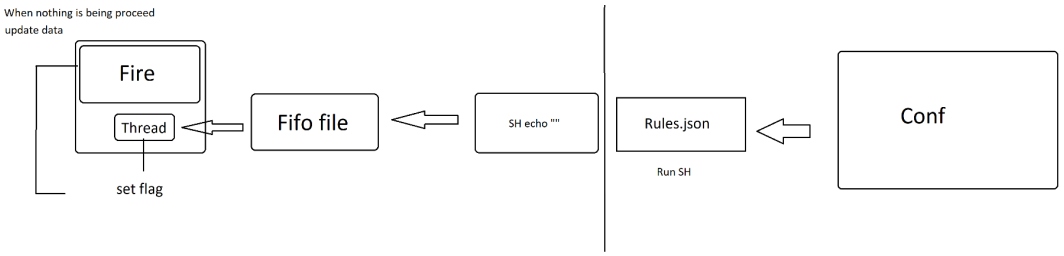
JĘZYKI PROGRAMOWANIA / FRAMEWORKI

Moduły Fire, Conf, emulatory serwerów ModBus i SLMP oraz testy jednostkowe zostały napisane w języku Python.
W projekcie jest również wykorzystywany potok nazwany - FIFO dostarczany przez większość systemów Uniksopodobnych.
Wykorzystano również framework Flask, do zbudowania aplikacji serwującej. Do definiowania wyglądu strony wykorzystano język HTML.

DIAGRAM KLAS

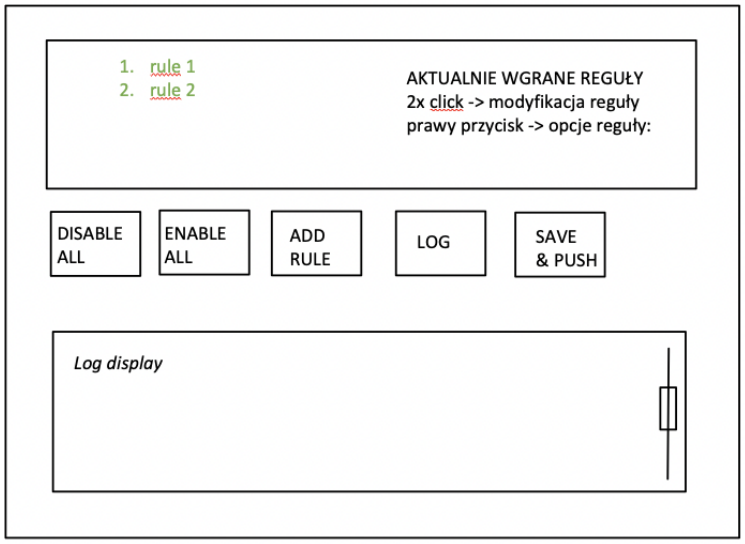


SCHEMAT DZIAŁANIA PROCESÓW WSPÓŁBIEŻNYCH



Moduł Conf po zapisie zdeterminowanych przez użytkownika zasad do pliku rules.json uruchamia skrypt zapisujący do FIFO, a wątek procesu Fire (czytający) zostaje uwolniony uruchamiając proces aktualizacji reguł.

“WIREFRAME” - PROTOTYP INTERFEJSU UŻYTKOWNIKA



Menu prawy przycisk myszy

disable	tymczasowo zablokuj regułę
edit	zmień regułę
delete	usuń regułę permanentnie
preview	

Disable all-> tymczasowe całkowite wyłączenie reguł przepuszczających pakiety. -> Odcień każdej z pozycji zmienia się na jaśniejszy jako symbol odznaczenia.

Enable All -> wszystkie odznaczone pozycje zmieniają kolor na ciemniejszy, domyślny, oznaczający aktywność

Add rule -> okno pop-up z polami do wprowadzania treści reguły.
OK -> pojawia się w oknie RULE LIST -> okno pop-up znika
CANCEL -> okno pop-up znika

Save rules -> okno popup weryfikujące użytkownika, następnie wpisane na nowo reguły zostają wgrane do systemu

Log - > powoduje wydrukowanie logu z 24h -> scroll to view

SPECYFIKACJA TESTÓW

SCENARIUSZE TESTÓW FUNKCJONALNYCH

W ramach testów funkcjonalnych modułu Fire przeprowadzono:

test_single_rule - sprawdzenie poprawności aktualizacji listy reguł po wywołaniu metody update_rules.
test_analyze_modbus_message_accept - test poprawności analizy pakietu ModBus **zgodnego** z wgranymi regułami
test_analyze_modbus_message_reject - test poprawności analizy pakietu ModBus **niezgodnego** z wgranymi regułami
test_analyze_modbus_packet_with_no_message_accept- test sprawdzający reakcję na pustą wadomość ModBus

test_compare_ip_tcp_with_rules_reject - test dla pakietów TCP **niezgodnych** z przykładową regułą
test_compare_ip_tcp_with_rules_accept - test dla pakietów TCP **zgodnych** z przykładową regułą

Specyfikacja analityczna

Model dziedziny

Dziedzina (wyłącznie moduł Fire): Sieć lokalna po kablach ethernetowych. Świat zewnętrzny na jednym interfejsie, świat broniony na drugim.

Dziedzina (pełna funkcjonalność): Powyższe + Interfejs sieciowy, może być bezprzewodowy do sieci lokalnej administratora udostępniający interfejs sieciowy konfiguracji reguł.

Słownik pojęć

Firewall

Zestaw skryptów pozwalający na regulowanie ruchu pomiędzy sieciami rozdzielonymi jednostką monitorującą. Projekt dotyczy monitorowania ruchu pakietów protokołów Modbus/TCP oraz SLMP.

Modbus/TCP i SLMP

Protokoły automatyki przemysłowej. Firewall poddaje przesyłane przez nie informacje analizie przez pryzmat reguł.

Protokół sieciowy

Zestandaryzowany sposób komunikacji pomiędzy jednostkami przez sieć.

Pakiet

Pojedynczy analizowany element przesyłanej między dwoma komputerami informacji. Element poddawany niezależnej analizie przez moduł Fire. Odrzucenie następuje gdy żadna z reguł przepuszczania nie jest aplikowalna dla danego pakietu.

Reguła

Pojedyncza instrukcja konfiguracyjna Firewalla. Definicja pożądanych cech pakietu, dzięki którym zostaje on zaakceptowany przez algorytm.

Interfejs sieciowy

Wejście przyjmujące pakiety z pojedynczej sieci. Wiąże się z osobną kartą sieciową i osobnym adresem IP. Może być np. ethernetowy (połączenie z siecią po kablu) czy bezprzewodowy (połączenie z siecią po WiFi).

Moduł

Pojedynczy skrypt pythonowy.

Podręcznik użytkownika

Wpisz w przeglądarkę adres IP RaspberryPi. Za pomocą interfejsu graficznego wybierz, dodaj bądź usuń reguły z Firewalla. Przesyłaj polecenia z stacji do jednostek wykonawczych i obserwuj działanie firewalla poprzez pojawiające się logi.

Budowa

Uruchom przez SSH na RaspberryPI oczekiwaną konfigurację reguł iptables:
sudo iptables -A INPUT -d {IP bronionego serwera w sieci ethernetowej} -j NFQUEUE

Następnie, uruchom moduły z pomocą poleceń:
sudo python3 Fire.py
sudo sh flask_docker/run_docker.sh

Konfiguracja

Stacje - współpraca i obsługa pakietów Modbus/TCP oraz SLMP.

RaspberryPi - wszystkie pakiety są już dostarczone i preinstalowane. W razie problemów, należy sięgnąć po następujące:
Flask==2.1.1, Docker==5.0.3, click==8.0.3, colorama==0.4.4, itsdangerous==2.0.1, Jinja2==3.0.3, MarkupSafe==2.0.1, Werkzeug==2.0.2, gunicorn==20.1.0, nfqueue==0.3.2, scapy==2.4.4.

Każdy z nich należy zainstalować za pomocą ***sudo pip3 install [nazwa pakietu]***.

Aktualizacja

Przed aktualizacją należy wyłączyć oba moduły. Następnie należy ściągnąć z repozytorium nową wersję kodu z gałęzi main i uruchomić ponownie, tak samo jak opisano powyżej, chyba że twórcy zaznaczą że należy postępować inaczej.

Poziomy dostępów

Użytkownicy mają trzy możliwe poziomy dostępu:
Zerowy: Są uczestnikami ruchu sieciowego w sieci lokalnej nadzorowanej przez Firewall, ale nie mają na niego wpływu.
Podstawowy: Interakcja z systemem jest ograniczona do interfejsu sieciowego. Znają hasło do interfejsu sieciowego i mogą konfigurować reguły Firewalla, nie mogą jednak go uruchamiać ani wyłączać.
Administracyjny: Interakcja z systemem jest możliwa poprzez RaspberryPi. Znają hasło do komputera i są zadeklarowanymi superużytkownikami.

Przechodzenie pomiędzy tymi stanami polega na udostępnianiu odpowiednich haseł użytkownikom.

Odtwarzanie systemu i kopie zapasowe

Przy nagłej potrzebie odtworzenia systemu należy pobrać ostatni działający plik rules.conf, w którym znajdują się wszystkie używane reguły firewalla i przeinstalować system, by następnie czysty plik rules.conf zastąpić swoje. Zaletą takiego rozwiązania jest niezaprzeczalnie przenaszalność i prostota.

Kopie zapasowe można wykonywać wsm tylko i wyłącznie z pliku konfiguracyjnego. Zalecanym jest tworzenie kopii zapasowej co godzinę - plik ten jest mały i nie powinien w istotny sposób zabierać miejsce.

Dokumentacja kodu źródłowego

Dokumentacja w formie Doxygen-a znajduje się na repozytorium pod adresem:
https://gitlab-stud.elka.pw.edu.pl/rstaszki/pzsp2-firewall