Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki

Rok akademicki 2013/2014

Praca dyplomowa inżynierska

Krzysztof Opasiak

Rozproszony monitoring systemów komputerowych

Opiekun pracy:
dr inż. Piotr Gawkowski

Ocena .....

.....
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Kierunek: Informatyka

Specjalność: Inżynieria Systemów Informatycznych

Data urodzenia: 1990.12.28

Data rozpoczęcia studiów: 2010.10.01

Życiorys

Urodziłem się 28 grudnia 1990 w Koninie. Uczęszczałem do Szkoły Podstawowej numer 8 im. Powstańców Wielkopolskich w Koninie. Następnie uczęszczałem do Gimnazjum Towarzystwa Salezjańskiego w Konienie.

W latach 2006-2010 uczęszczałem do Technikum w Zespole Szkół im. Mikołaja Kopernika w Koninie. W trakcie nauki w tej szkole dwukrotnie przyznano mi stypendium Prezesa Rady Ministrów za bardzo dobre wyniki w nauce oraz wzorowe zachowanie. W roku 2010 ukończyłem z wyróżnieniem szkołę średnią, a następnie zdałem maturę oraz egzamin zawodowy uzyskując tytuł Technik Teleinformatyk.

W październiku 2010 roku rozpocząłem studia stacjonarne pierwszego stopnia na Wydziale Elektroniki i Technik Informacyjnych na kirunku Informatyka.

.....
podpis studenta

Egzamin dyplomowy

Złożył egzamin dyplomowy w dn. .....20__r

Z wynikiem .....

Ogólny wynik studiów .....

Dodatkowe wnioski i uwagi Komisji .....

.....

Streszczenie

Praca ta prezentuje . . .

Słowa kluczowe: *słowa kluczowe.*

Abstract

Title: *Thesis title.*

This thesis describes . . .

Key words: *key words.*

Spis treści

1. Wprowadzenie

Komputer stał się nieodzowną częścią współczesnej kultury. Praktycznie każde gospodarstwo domowe posiada komputer wraz z dostępem do internetu. Urządzenia będące w posiadaniu prywatnych właścicieli, bardzo często są wykorzystywane do rozrywki lub innych czynności, których niewykonanie nie pociąga za sobą żadnych konsekwencji. Znaczna jednak część urządzeń znajduje się w posiadaniu dużych firm oraz ośrodków badawczych. Komputery te zazwyczaj są połączone ze sobą w sieć prywatną - intranet. Do ich połączenia konieczna jest zarówno rozbudowana struktura okablowania jak i zestaw urządzeń sieciowych. Wykorzystanie komputerów pozwala firmom na przyśpieszenie prac, przez co ich zysk znacząco wzrasta. Brak możliwości używania komputera, lub komunikacji poprzez infrastrukturę sieciową, niesie za sobą poważne straty finansowe. Konieczne jest zatem zapewnienie funkcjonowania całej infrastruktury, w każdej chwili, gdy jest ona potrzebna.

Urządzenia elektroniczne posiadają ograniczoną trwałość, przez co istnieje niezerowe prawdopodobieństwo awarii każdego z elementów sieci firmowej. Ponadto należy pamiętać, iż na urządzeniach uruchamiane jest oprogramowanie, które może zawierać błędy. Za awarię w danym systemie należy zatem uznać, nie tylko fizyczne uszkodzenie urządzenia, lecz także sytuacje, w której użytkownik zostaje pozbawiony dostępu do danej aplikacji. Warto również zauważyć, że użytkownik oczekuje od danej aplikacji dostarczenia usług o odpowiednim poziomie. Zatem należy również uznać, za awarię, sytuację, gdy usługi świadczone użytkownikowi nie są na satysfakcjonującym poziomie.

Użytkownik końcowy bardzo często nie jest w stanie udzielić precyzyjnej informacji o usterce, której doświadczył. Bardzo popularna jest sytuacja, w której użytkownik zgłasza, że nie działa aplikacja z której korzysta, natomiast faktyczną przyczyną błędu jest awaria bazy danych lub komunikacji sieciowej. Indywidualna diagnoza przy każdej awarii jest czasochłonna, przez co czas do jej usunięcia wydłuża się, powodując straty finansowe. Od wielu lat w celu optymalizacji wykrywania i obsługi awarii stosuje się systemy monitorujące, które przedstawiają administratorowi w przejrzysty sposób stan wszystkich urządzeń oraz usług.

Zadaniem systemu monitorującego jest śledzenie stanu danego urządzenia i przedstawianie go administratorowi poprzez czytelny interfejs użytkownika. Stan urządzenia może być rozumiany bardzo szeroko. Istnieją systemy, które pozwalają na sprawdzanie, nie tylko czy urządzenie jest włączone, lecz również jego szczegółowych parametrów takich jak temperatura poszczególnych podzespołów czy zużycie prądu. Można wyróżnić dwa podstawowe rodzaje monitorowania:

Monitorowanie aktywne rodzaj monitorowania, w którym system monitorujący cyklicznie wykonuje sprawdzenie stanu danego urządzenia lub usługi

Monitorowanie pasywne rodzaj monitorowanie, w którym status usługi lub urządzenia zgłaszany jest przez program zewnętrzny do systemu monitorującego

Każdy ze sposobów monitorowania ma zarówno swoje wady i zalety. Wybór metody monitorowania, zależy zatem od charakterystyki monitorowanej wartości. Jeśli parametr podlega, nieregularnym i krótkotrwałym zmianom, a każda z nich powinna być odnotowana stosuje się monitorowanie pasywne. Natomiast jeśli dana wartość posiada charakterystykę zmieniającą się w sposób ciągły, należy korzystać wtedy z monitorowania aktywnego, które dokonuje próbkowania danej wartości w określonych odstępach czasu.

Sieci bardzo dużych przedsiębiorstw, posiadają budowę wielosegmentową. Ze względów bezpieczeństwa bardzo często składa się ona z sieci wirtualnych, czy wręcz odizolowanych od siebie podsieci. Ponadto sieci przedsiębiorstwa bardzo często zawierają zapory ogniowe, które filtrują ruch pomiędzy sieciami. Ze względu na fragmentacje sieci często nie jest możliwe zastosowanie prostego systemu monitorowania opisanego wcześniej. Konieczne jest zatem użycie systemu rozproszonego.

Najprostszą realizacją rozproszonego systemu monitorowania jest użycie monitorowania pasywnego do monitorowania wszystkich urządzeń i usług, które nie są widoczne z sieci w której uruchomiony jest system monitorujący. Niestety wymaga to zmian w konfiguracji wszystkich urządzeń i uruchomienia na nim dodatkowego oprogramowania. Takie zmiany mogą nie być możliwe, na prostych urządzeniach, których kontrola odbywa się poprzez preinstalowany system producenta.

Możliwa jest również konfiguracja, wieloinstancyjnego systemu monitorowania. W każdej odizolowanej komórce sieci należy umieścić instancję systemu, która będzie zbierała dane z tej komórki sieci. Monitorowanie danego fragmentu infrastruktury może się odbywać zarówno w sposób aktywny jak i pasywny. Po wstępnym przetworzeniu takich danych, muszą one zostać zsynchronizowane pomiędzy instancjami, a następnie umieszczone w instancji nadrzędnej lub innym zbiorczym miejscu docelowym. Rozwiązanie to posiada liczne zalety i jest bardzo często stosowane. Dodatkowo niektóre z systemów umożliwiają wymianę danych pomiędzy instancjami bez konieczności istnienia wyróżnionej instancji nadrzędnej.

Przedstawienie administratorowi danych bieżących jest często niewystarczaje. Precyzyjna diagnoza awarii w możliwe krótkim czasie od jej wystąpienia jest bardzo ważna. Jednak istotna jest również możliwość analizy historycznych awarii, aby umożliwić wykrycie potencjalnej awarii jeszcze przed jej wystąpieniem. Dostępne są na rynku systemy, które pozwalają na gromadzenie danych o odczytach w bazach danych. Kolejnym krokiem może być analiza takiej bazy danych z wykorzystaniem systemu eksperckiego, który wykaże odpowiednie zależności i na tej podstawie umożliwi wykrycie potencjalnej awarii jeszcze przed jej wystąpieniem.

Współczesne korporacje posiadają nie tylko rozbudowaną infrastrukturę sieciową, lecz również bardzo dużą liczbę urządzeń mobilnych takich jak laptopy, tablety czy inne urządzenia specyficzne dla danej firmy. Bardzo często okazuje się, że poprawne działanie tych urządzeń wpływa znacząco na efektywność pracy osób, które ich używają. Monitorowanie takiego urządzenie jest zadaniem nietrywialnym. Należy pamiętać, iż urządzenie mobilne może nie mieć chwilowej możliwości komunikacji z systemem monitorującym. Jeśli przerwy w łączności występują stosunkowo często, to w przypadku braku późniejszej synchronizacji danych można doprowadzić, do fałszywych predykcji systemu eksperckiego. Aby tego uniknąć konieczne jest zapewnienie dostarczenia wyników wszystkich pomiarów do systemu, kiedy tylko stanie się to możliwe. W przypadku klienta mobilnego niezwykle istotna jest również kwestia bezpieczeństwa. Urządzenia takie często nie pracują wewnątrz sieci formowej, lecz używają wielu różnych, niezaufanych sieci do komunikacji.

Dane zebrane podczas monitorowania klienta mobilnego mogą zawierać tajemnice handlowe firmy. Konieczne jest zatem zapewnienie zarówno poufności jak i integralności danych podczas synchronizacji.

Niestety, obecnie na rynku brak jest rozwiązań, które umożliwiałyby monitorowanie klienta mobilnego. Ważne jest dostarczenie odpowiedniego systemu, który pozwoli na kompleksowe monitorowanie wszystkich urządzeń występujących w firmie, zarówno mobilnych jak i statycznych. W związku z powyższym w niniejszej pracy wykonano rozbudowę popularnego systemu monitorowania, aby umożliwić monitorowanie przy jego użyciu zarówno urządzeń statycznych jak i mobilnych.

Układ tej pracy jest następujący. Rozdział 2 zawiera opis oraz porównanie dostępnych na rynków systemów monitorowania. W rozdziale 3 przedstawiono problematykę monitorowania klienta statycznego oraz mobilnego. Ponadto po wykonaniu analizy, przedstawiono wymaganie jakie są stawiane przed systemem kompleksowego monitorowania przedsiębiorstwa. Rozdział 4 zawiera opis szczegółowy opis systemu Icinga na bazie którego budowany jest projektowany system. W rozdziale 5 przedstawiono projekt systemu monitorowania, zgodnego z przedstawionymi wymaganiami. Rozdział 6 zawiera natomiast opis wykonanej w ramach niniejszej pracy implementacji systemu monitorowania. W rozdziale 7 zawarto opis przebiegu testowania wykonanego systemu, a także przedstawiono sprawozdanie z jego użytkowania. Rozdział 8 natomiast, zawiera podsumowanie niniejszej pracy, a także wskazuje potencjalne możliwości rozwoju wykonanego systemu.

2. Dostępne systemy monitorujące

2.1. Przegląd systemów dostępnych na rynku

Na rynku dostępnych jest wiele bardzo różnych systemów monitorujących. Narzędzia z tej grupy możemy podzielić na dwie kategorie:

— Systemy dostępnościowe,
— Systemu analityczne.

Systemy monitorujące w których główny nacisk położony jest na zapewnienie ciągłej dostępności monitorowanych usług nazywane są systemami dostępnościowymi. Wspierają one administratora w codziennych zadaniach, poprzez nieustanne monitorowanie aktualnego stanu sieci. Narzędzia te są wykorzystywane przede wszystkim do szybkiego powiadamiania oraz lokalizacji awarii.

Systemy analityczne, w kontekście monitorowania infrastruktury sieciowej, to systemy, które są nastawione na zbieranie i analizę posiadanych danych. Tego typu systemy nie są zazwyczaj wykorzystywane do powiadamiania czy lokalizacji awarii. Ich zadaniem jest przede wszystkim gromadzenie danych dotyczących zużycia poszczególnych zasobów, czy też wskaźników jakości poszczególnych usług. Systemy te posiadają zazwyczaj bardzo rozbudowane narzędzia służące do generacji i analizy wykresów na podstawie zebranych wcześniej danych.

W ostatnich latach można zauważyć wzrost popularności rozwiązań hybrydowych. Pozwalają one na kompleksowe zarządzanie infrastrukturą sieciową. Dzięki zastosowaniu takiego systemu administrator uzyskuje jeden uniwersalny interfejs. Możliwy jest w nim zarówno pogląd bieżącego stan sieci oraz diagnoza awarii, jak i analiza danych historycznych.

Przechowywanie danych zgromadzonych podczas monitorowania może odbywać się na różne sposoby. Podstawową techniką przechowywania danych, jeszcze 5 lat temu były płaskie struktury pliki zawierające zgromadzone dane. Rozwiązanie tego typu jest bardzo uciążliwe, a sprawne zarządzanie zgromadzonymi danymi wymaga dużego wkładu pracy własnej administratora. Obecnie rozpowszechniają się techniki przechowywania zebranych danych w oparciu o bazy danych. Współcześnie używane typy baz danych to:

— Relacyjne bazy danych
— Cykliczne bazy danych[1]

Dane przechowywane w relacyjnych bazach danych zorganizowane są w postaci tabel, a powiązania pomiędzy danymi nazywane są relacjami. Taka organizacja bazy danych sprawia, że wraz z upływem czasu jej rozmiar rośnie. Powoduje to zwiększenie zajętości przestrzeni dyskowej, a także wpływa na czas wykonywania

[1] ang. *Round Robin Database*

operacji. Dane są przechowywane w bazie do czasu, gdy użytkownik jawnie je usunie. Pozwala to na przeglądanie dowolnie długiego okresu historii, bez utraty dokładności, a także na dynamiczne zarządzanie czasem przechowywania danych. Narzędzia korzystające z tego typu baz muszą posiadać pewną politykę zarządzania zgromadzonymi danymi. Istotne jest jednak, że polityka ta jest uzależniona jedynie od aplikacji, a nie od samej bazy danych i może być zmieniana w dowolnym momencie.

Cykliczne bazy danych posiadają natomiast stały, definiowany podczas tworzenia rozmiar. Rozmiar ten określa liczbę porcji danych jaka może być przechowywana w bazie. Jeśli rozmiar bazy przekroczy rozmiar zadany przy tworzeniu, wykonywana jest konsolidacja danych. Polega ona na wyliczeniu zadanych wartości w odpowiednich przedziałach i zachowanie ich w pojedynczych rekordach, oraz usunięcie dokładnych danych. Możliwe są trzy typy konsolidacji danych, minimum, średnia oraz maksimum. Rozmiar bazy danych jest definiowany w chwili jej tworzenia i późniejsza jego modyfikacja nie jest już możliwa. Ponadto należy zwrócić szczególną uwagę, na fakt iż dane są usuwane z bazy danych bez wiedzy użytkownika czy aplikacji, przez co taka baza danych nie może zostać użyta do dokładnej analizy danych historycznych.

Każdy typ systemu, jak i rodzaj bazy danych posiada swoje zastosowanie. Należy zatem rozważnie zdefiniować wymagania jakie stawia się przed systemem. Precyzyjne sformułowanie wymagań oraz dokładna analiza możliwości każdego z systemów zapewni wybór narzędzia dostosowanego do potrzeb. Możliwa jest jednak sytuacja, w której żaden z dostępnych systemów, nie będzie spełniał wszystkich stawianych wymagań. Konieczne jest w takiej sytuacji podjęcie wysiłku modyfikacji systemu spełniającego najwięcej wymagań, lub zaprojektowanie oraz implementacja takiego systemu od podstaw.

2.1.1. System monitorowania Cacti

Jest to system monitorujący, rozwijany przez The Cacti Group Inc. i dystrybuowany na licencji GPL[2]. System bazuje na oprogramowaniu RRDtool. Jest to narzędzie, które pozwala na wykorzystanie cyklicznej bazy danych do składowania pomiarów wartości w zadanym przedziale czasowym. Ponadto dostarcza ono funkcji do generacji wykresów w kilku formatach. Dokładny opis wszystkich możliwości RRDTool można znaleźć w [13]. Dzięki wykorzystaniu wspomnianego narzędzia system ma bardzo prostą budowę i składa się z następujących elementów:

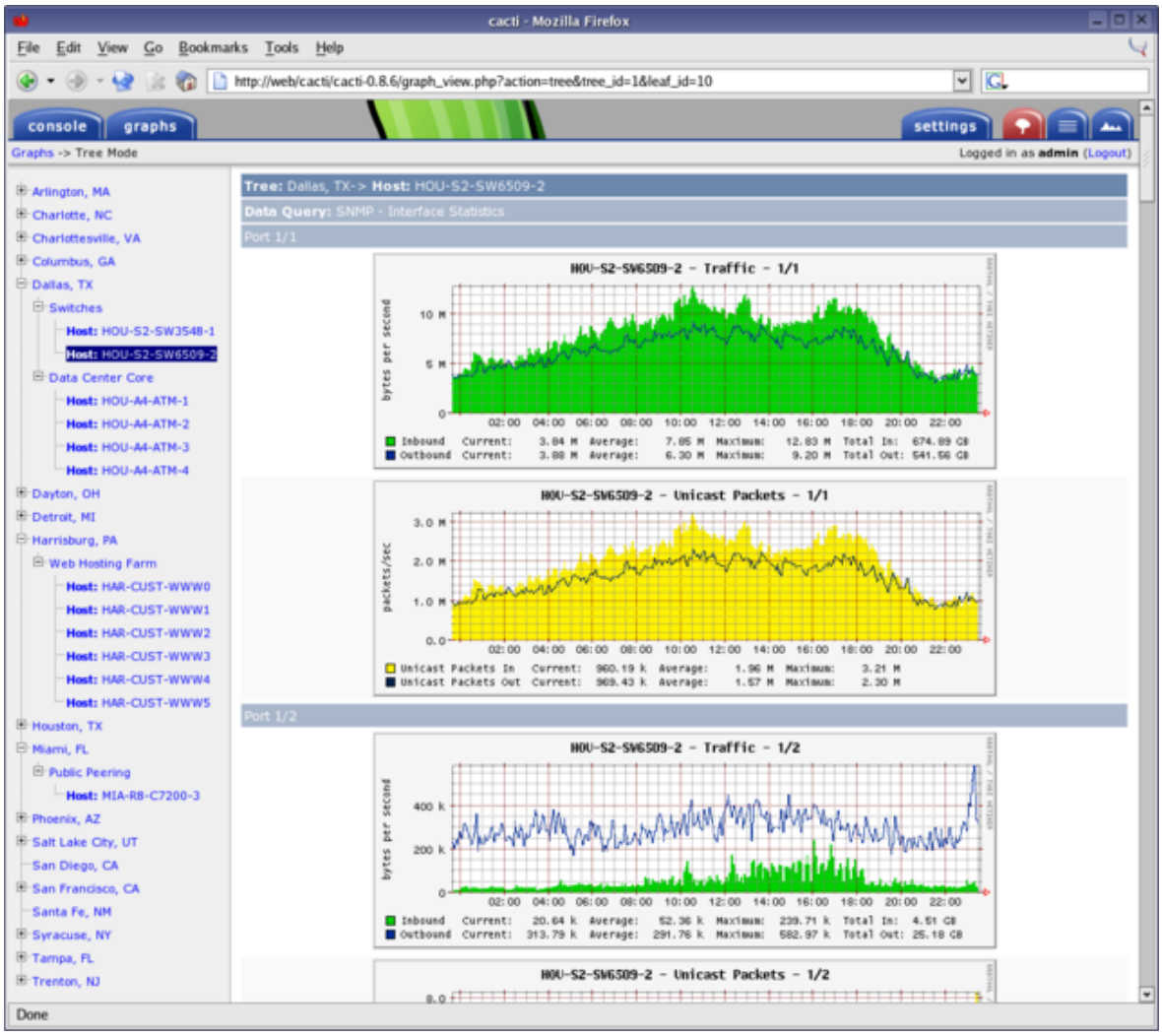
— interfejs użytkownika,
— dostawca danych.

Interfejs użytkownika został napisany w języku PHP. Do jego działania niezbędny jest serwer http np. Apache. Rysunek 2.1.1 przedstawia przykładową podstronę systemu Cacti. Z poziomu interfejsu użytkownika możliwa jest graficzna konfiguracja całego systemu. Interfejs posiada klasyczna budowę. Składa się on z jednokolorowego paska menu, w którym zawarte są odnośniki do poszczególnych podstron oraz z pulpitu, na którym wyświetlane są wybrane dane. Interfejs umożliwia graficzne przedstawienie wyników w postaci wykresów. Format wykresu może być definiowany bezpośrednio przez użytkownika, lub można skorzystać z bogatej

[2] ang. *General Public License* - popularna licencja oprogramowania o otwartych źródłach. Treść licencji można znaleźć w [12].

biblioteki gotowych szablonów. Dostęp do interfejsu zabezpieczony jest poprzez mechanizm uwierzytelnienia użytkownika systemu monitorującego. Możliwe jest definiowanie wielu użytkowników oraz ich uprawnienia. Każdy użytkownik ma możliwość definiowania własnego zestawu wykresów oraz pulpitów.

Rysunek 2.1. Interfejs użytkownika systemu Cacti.



Dostawca danych jest to element systemu, który jest odpowiedzialny za faktyczne wykonywanie sprawdzeń danej wartości i przekazywanie ich do narzędzia RRDTool. System umożliwia wybór jednego z dwóch dostawców danych. Pierwszym z nich jest cmd.php, który jest prostym skryptem napisanym w języku PHP. Umożliwia on monitorowanie aktywne urządzeń przy pomocy protokołu SNMP[3]. Skrypt cmd.php przeznaczony jest do monitorowania jedynie niewielkich sieci. Ze względów wydajnościowych, nie jest możliwe wykorzystanie go do monitorowania rozległej infrastruktury.

[3] *Simple Network Management Protocol* – protokół zarządzania urządzeniami sieciowymi i uzyskiwania informacji o ich stanie. Zorganizowany w formie drzewa, gdzie każdy liść posiada globalnie unikalny identyfikator o ściśle określonym znaczeniu. Szeroko opisany w [10].

Drugim z możliwych do wyboru dostawców danych jest narzędzie Spine, nazywany również Cactid. Jest to program napisany w języku C, który uruchomiony jest jako serwis systemowy na urządzeniu monitorującym. Umożliwia on monitorowanie urządzeń zarówno poprzez protokół SNMP jak i z wykorzystaniem innych metod. Możliwość dostarczenia własnych metod monitorowania opiera się na dostarczeniu skryptu lub pliku wykonywalnego, który będzie cyklicznie uruchamiany przez Cactid, a jako wyniki przekazywane w taki sam sposób jak z sprawdzeń opierających się na SNMP.

Żaden z dostawców danych nie umożliwia monitorowania danego urządzenia lub usługi w sposób pasywny. Cacti nie posiada również żadnego mechanizmu, który pozwoliłby na monitorowanie sieci w sposób rozproszony. Oznacza to, iż administrator musi zmienić konfigurację sieci, tak aby jeden serwer miał dostęp do każdego urządzenia, lub konfigurować i zarządzać osobną instancją w każdym segmencie. Jest to bardzo niewygodne i wręcz uniemożliwia monitorowania rozległych sieci przy pomocy Cacti[4].

2.1.2. System monitorowania Nagios

System Nagios został opublikowany w 1999 na licencji GPL. System od niemal 15 lat jest ciągle rozwijany i udoskonalany, zarówno przez autorów jak i przez szeroką społeczność. W systemie Nagios najwyższym priorytetem jest dbałość o zapewnienie dostępności wszystkich monitorowanych usług. Organizacja systemu zakłada, iż w sieci znajdują się urządzenia, które mogą świadczyć pewne usługi. Każde urządzenie jak i usługa może być w jednym z trzech stanów logicznych:

OK usługa działa poprawnie
WARNING monitorowane parametry przekroczyły stan ostrzegawczy
CRITICAL parametry usługi przekroczyły stan krytyczny, usługa lub urządzenie nie funkcjonuje

System posiada rozbudowane algorytmy określania stanu każdego urządzenia oraz usługi. Działanie usługi, jest zawsze zależne od stanu urządzenie, na którym dana usługa jest świadczona. Ponadto użytkownik może definiować zależności pomiędzy urządzeniami. System Nagios posiada rozbudowany system powiadamiania administratora o wystąpieniu awarii oraz o jej zakończeniu, lub innych zdefiniowanych wydarzeniach systemowych. Ponadto możliwe jest automatyczne wykonywanie programów lub skryptów, jeśli wystąpiło jakieś zdarzenie. Podstawowa wersja systemu składa się z następujących elementów:

— Interfejs graficzny
— Rdzeń monitorujący

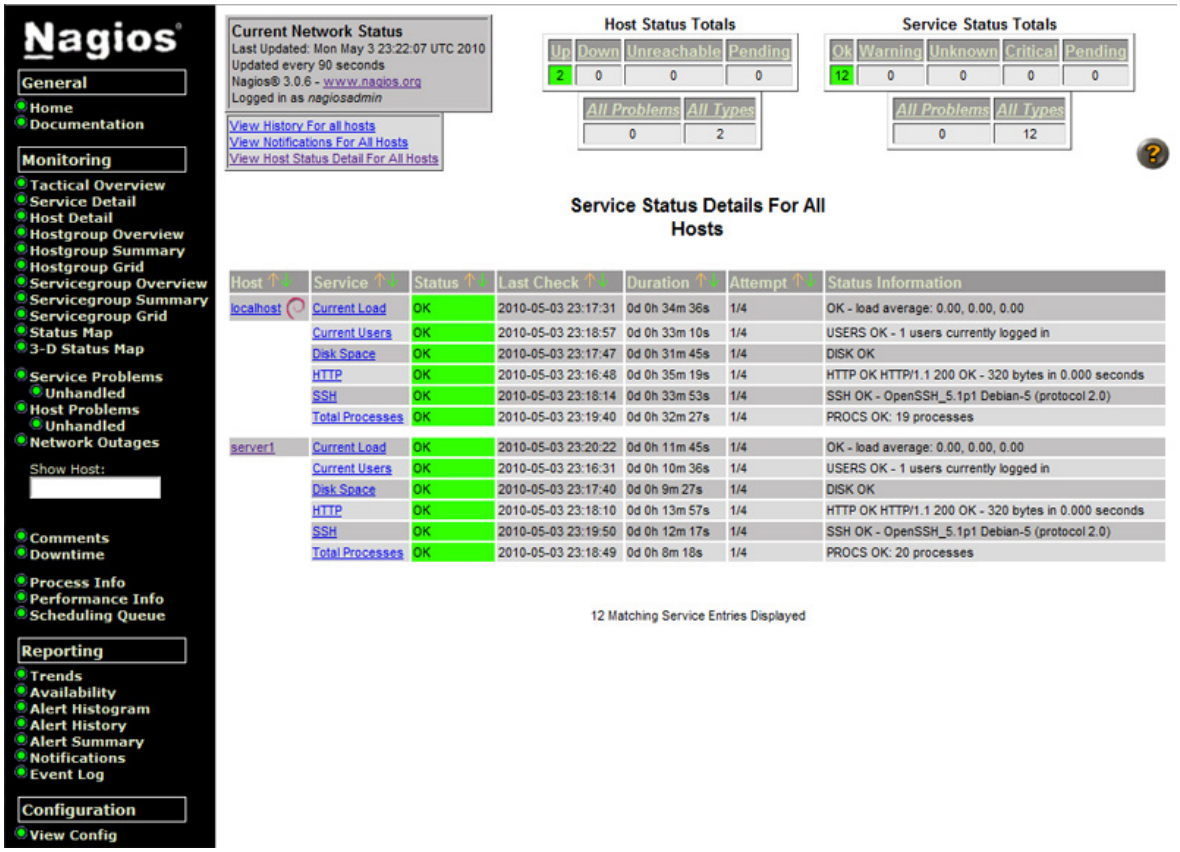
Interfejs graficzny został napisany w języku C z wykorzystaniem technologi CGI[5]. Jego wygląd jest zgodny z standardami z lat 90. Klasyczna strona WWW bez dynamicznie zmieniającej się treści. Dane odświeżane są na żądanie klienta, lub co określony czas. Wykorzystana technologia zakłada przesyłanie za każdym

[4] Więcej informacji na temat systemu monitorowania Cacti można znaleźć w [3].

[5] *Common Gateway Interface* – znormalizowany interfejs służący do komunikacji pomiędzy serwerem www, a zewnętrznymi programami. Interfejs ten jest wykorzystywany do generowania stron internetowych na żądanie klienta. Zewnętrzny program generuje stronę w języku HTML, a następnie serwer przesyła ją do klienta poprzez serwer http. Szczegółowy opis można znaleźć w [4]

razem całego dokumentu HTML do klienta, w związku z czym generowany jest nadmierny ruch sieciowy. Widok użytkownika składa się z kliku części. Po lewej stronie widoczne jest klasyczne menu, umożliwiające użytkownikowi wybór treści. Na górze strony natomiast znajduje się podsumowanie aktualnego stanu monitorowanych urządzeń i usług. Centralną część okna zajmuje pulpit, który prezentuje użytkownikowi treść wybraną wcześniej z menu. Interfejs użytkownika umożliwia podgląd aktualnego stanu usług oraz urządzeń. Informacja ta może być wyświetlana w formie listy zawierającej urządzenia i usługi, lub w postaci mapy sieci, która pozwala na monitorowanie stanu urządzenia w korelacji z jego logicznym umieszczeniem w strukturze sieciowej. Możliwe jest również przeglądanie historii awarii oraz prostych wykresów stanu urządzenia lub usługi w zadanym przedziale czasu. Dostęp do interfejsu chroniony jest przy pomocy autoryzacji http. Możliwe jest definiowanie wielu użytkowników, jednak tylko z poziomu urządzenia na którym uruchomiony jest system monitorujący. Należy zauważyć również, że wszyscy użytkownicy danego typu posiadają takie same uprawnienia. Nie ma możliwości dowolnej edycji uprawnień danej grupy czy też użytkownika.

Rysunek 2.2. Interfejs użytkownika systemu Nagios.



Rdzeń monitorujący został zaimplementowany w języku C. Jest to centrum całego systemu, gdyż zajmuje się on przetwarzaniem wszystkich bieżących danych monitorowania, a następnie składowaniem ich w plikach. Ta cześć systemu jest odpowiedzialna za wykonywanie sprawdzeń w określonych odstępach czasu. Każde

sprawdzenie odbywa się poprzez wykonanie komendy zdefiniowanej przez użytkownika. Komenda ta może zawierać zarówno wykonanie pliku binarnego jak i dowolnego skryptu. W ramach projektu Nagios, rozwijany jest zestaw wtyczek[6], czyli programów służących do monitorowania podstawowych usług oraz parametrów urządzeń. Dostępna jest bardzo duża liczba wtyczek, dzięki czemu system Nagios może monitorować w sposób aktywny wszystkie podstawowe parametry lub usługi. Możliwe jest również definiowanie własnych wtyczek, które będą sprawdzały stany pewnych specyficznych dla danej sieci parametrów. W celu zapewnienia poprawnego funkcjonowania całego systemu, konieczne jest, aby dodatkowe wtyczki spełniały wymagania opisane w [8]. System umożliwia również monitorowanie dowolnych usług w sposób pasywny. Programy dostarczające odczytów pasywnych również są zobowiązane do przestrzegania wcześniej wspomnianych zasad.

System posiada rozbudowane możliwości monitorowania rozproszonego. Niestety, do wykonania znacznej części z tych konfiguracji potrzebne są elementy systemu, które są dystrybuowane na licencjach komercyjnych wymagających zakupu praw do korzystania z nich. Istnieją również darmowe dodatki, które pozwalają na przechowywanie zgromadzonych danych zarówno w bazie relacyjnej jak i cyklicznej. Możliwa jest również częściowa integracja systemu Nagios[7] z dodatkami lub systemami, które pozwalają na wizualizacje zgromadzonych danych.

2.1.3. System monitorowania Icinga

System Icinga powstał w 2009 roku jako klon (ang. *fork*) systemu Nagios. System został wzbogacony o wiele nowych elementów, a także poprawiono wiele błędów obecnych w systemie Nagios. Dzięki zachowaniu wstecznej kompatybilności zarówno wszystkie wtyczki jak i dodatki systemu Nagios mogą być wykorzystane w systemie Icinga. Pozyskano dzięki temu bardzo dużą bazę wtyczek, co umożliwia monitorowanie tych samych usług i urządzeń co przodek.

System Icinga został wyposażony w zupełnie nowy interfejs graficzny[8]. Został on zaimplementowany w języku PHP przy użyciu szkieletu aplikacji agavi[9]. Jest on zatem oparty na technologii Ajax, dzięki której komunikacja z użytkownikiem, nie opiera się na przesyłaniu całych stron w języku HTML, lecz na realizacji żądań generowanych poprzez język skryptowy wykonywany po stronie użytkownika. Dzięki zastosowaniu tej technologi, proces wyświetlania strony zużywa mniejszą część pasma, a serwer został odciążony. Nowy interfejs użytkownika jest w pełni dynamiczny, składa się on z rozszerzalnego menu po lewej stronie oraz pulpitów użytkownika w centralnej części. Możliwe jest otwieranie wielu pulpitów oraz wyświetlanie poszczególnych informacji w osobnych oknach, które można swobodnie przemieszczać w obszarze strony. Przykładowy pulpit dostępny dla użytkownika został przedstawiony na **??**.Znacznej zmianie uległ również model bezpieczeństwa. W nowym interfejsie graficznym, każdy użytkownik, posiada swój zestaw zdefiniowanych uprawnień. Oznacza to, że możliwe jest ograniczenie użytkownikowi

[6] Należy zwrócić uwagę na różne znaczenie słów wtyczka (ang. *Plugin*) oraz dodatek (ang. *Addon*).

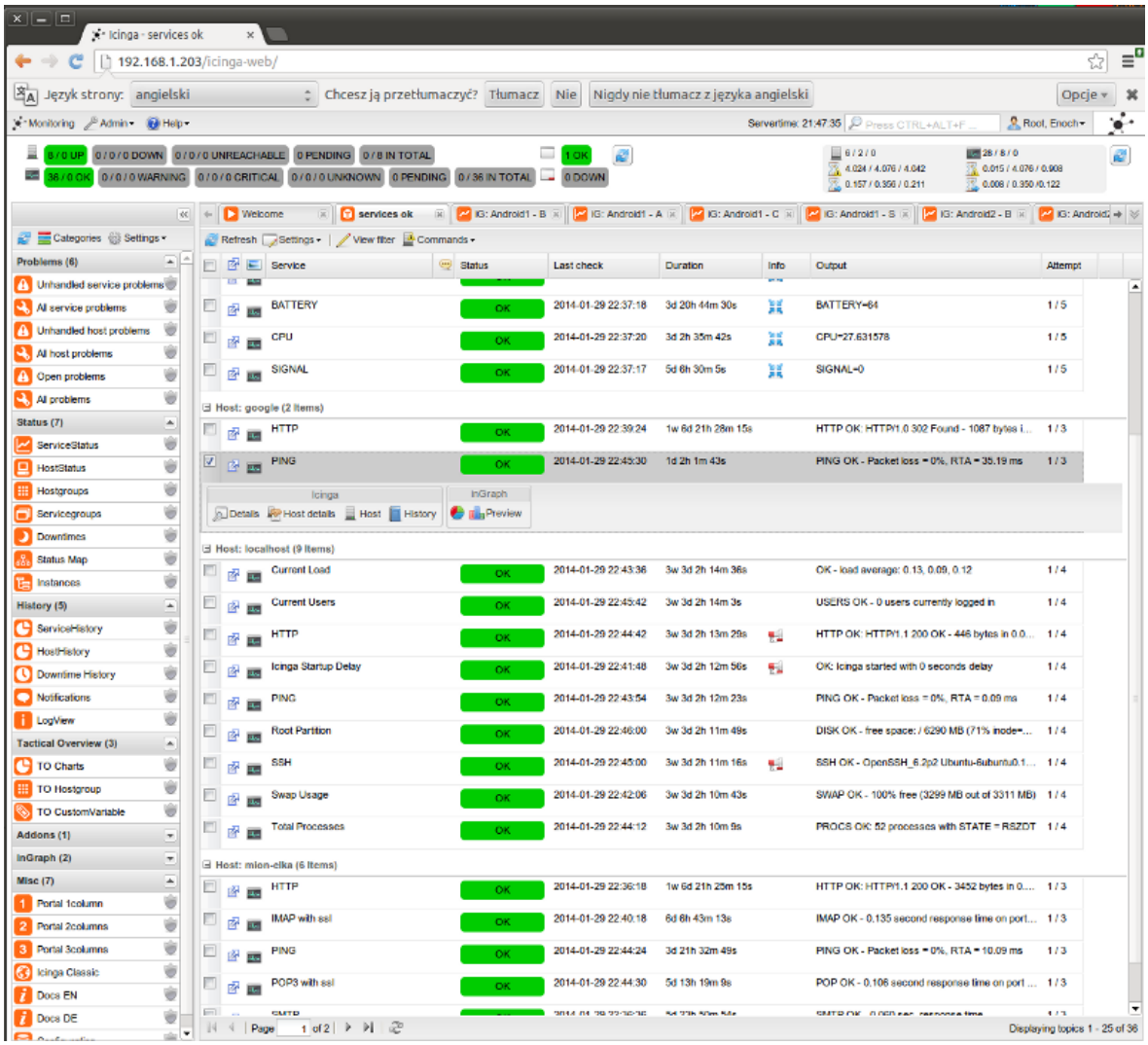
[7] Szczegółowe informacje na temat rozszerzania funkcjonalności oraz samego systemu można znaleźć w [9].

[8] Skorzystanie z nowego interfejsu wymaga użycia modułu IDOUtils oraz bazy danych. Możliwe jest wykorzystanie również klasycznego interfejsu, który nie posiada takich wymagań.

[9] *Agavi* – szkielet aplikacji języka PHP5 pozwalający na łatwą realizację funkcjonalności zgodne z modelem programowym Model-Widok-Kontroler. Szerszy opis znajduje się na stronie domowej projektu: [1]

dostępu do danych o konkretnej usłudze lub zabronić wykonywania niektórych czynności administracyjnych. Zarządzanie użytkownikami oraz ich uprawnieniami możliwe jest również z poziomu graficznego interfejsu użytkownika, co znacząco podnosi wygodę użytkowania systemu.

Rysunek 2.3. Interfejs użytkownika systemu Icinga.



Kolejną istotną różnicą, jest zmiana architektury systemu. System Nagios posiada budowę monolityczną, a współpraca pomiędzy poszczególnymi jego komponentami odbywa się w sposób bardzo zawiły i niejednorodny. System Icinga wprowadził natomiast budowę modularną. Wszystkie możliwe komponenty systemu zostały wyodrębnione, a do swobodnej komunikacji pomiędzy nimi zdefiniowano wygodne API. Taka budowa umożliwia przede wszystkim rozmieszczenie poszczególnych komponentów systemu na różnych fizycznych maszynach, co w przypadku dużych sieci może spowodować znaczący wzrost wydajności i niezawodności. Dostarczenie jednolitego REST API[10] umożliwia również prostsze tworzenie dodatków

[10] ang. *Representational state transfer* – lekka metoda przesyłania danych pomiędzy klientem a serwerem.

rozbudowujących możliwości systemu. W systemie Icinga rozbudowano także możliwości współpracy z bazą danych. System ten umożliwia współpracę, już nie tylko z bazą MySQL, lecz również z bazami PostgreSQL czy też z systemem zarządzania bazą danych firmy Oracle. Możliwość wykorzystania bazy danych Oracle, jest bardzo istotna, jeśli dane dotyczące konfiguracji muszą być przechowywane przez bardzo długi czas, lub jeśli monitorowana infrastruktura jest bardzo rozbudowana.

System Icinga, nie tylko umożliwia rozmieszczenie modułów na różnych fizycznych maszynach, lecz również umożliwia wiele innych konfiguracji, które można wykorzystać podczas monitorowania rozproszonego. Szczególnie wartą zauważenia jest konfiguracja, w której występuje wiele równorzędnych instancji rdzenia monitorującego, natomiast wszystkie współpracują używając jednej bazy danych. Centralna baza danych stanowi źródło danych dla interfejsu graficznego. Taka konfiguracja umożliwia monitorowanie bardzo rozległej lub wielosegmentowej infrastruktury i prezentacji wyników poprzez jeden interfejs. Należy również nadmienić, iż wszystkie elementy niezbędne do konfiguracji takiego rozwiązania są darmowe.

2.2. Podsumowanie

Współczesne systemy monitoringu, są bardzo bogato wyposażone i posiadają szereg zaawansowanych możliwości. Każdy z systemów oferuje unikalny zestaw rozwiązań, które z pewnością mogą zostać wykorzystane w wielu instytucjach. Porównując wszystkie omówione systemy, należy zwrócić szczególną uwagę, na różnice w ich możliwych zastosowaniach docelowych.

Systemy, takie jak Cacti zaliczane są do grupy systemów analitycznych. Ich celem jest zatem zapewnienie możliwości gromadzenia oraz analizy danych. Zbierane dane mają charakter zagregowany w zadanych przedziałach, na podstawie których prezentowane są użytkownikowi odpowiednie wykresy. Niestety ze względu na główny sposób gromadzenia danych - protokół SNMP, oraz ubogość innych metod, systemy te nie mogą być wzbogacone o funkcjonalność charakterystyczną dla systemów dostępnościowych.

Drugą grupę systemów stanowią natomiast systemy dostępnościowe, takie jak Nagios czy Icinga. Ich głównym celem jest monitorowanie bieżącego stanu infrastruktury i raportowanie użytkownikowi najświeższych informacji. Systemy te zostały również zaprojektowane, aby wspomagać administratora w lokalizacji awarii. Głównym typem danych na których operują te systemy jest stan urządzenia lub usługi. Zdefiniowanie odpowiednich poziomów kwantyzacji dla stanów pozwala na szybkie uzyskiwanie poglądowych informacji o stanie sieci. Podczas monitorowania gromadzone są również dane szczegółowe. Ich przetwarzaniem nie zajmują się już jednak same systemy monitorowania, lecz liczne dodatki do nich. Możliwe jest zatem rozbudowanie systemu tego typu, o dodatkowe elementy, które pozwolą uzyskać system hybrydowy. System taki będzie mógł pełnić rolę zarówno systemu dostepnościowego jak i analitycznego.

Wybierając system monitorujący, należy zatem dokonać szczegółowej analizy wymagań stawianych przed systemem. Szczegółowe porównanie wszystkich przedstawionych systemów monitorowania zawarto w 2.1.

Tablica 2.1: Porównanie systemów monitorowania

Nazwa systemu	Cacti	Nagios	Icinga
Podgląd stanu bieżącego	Nie	Tak	Tak
Podgląd danych historycznych	Tak	Tak, przez dodatek	Tak, przez dodatek
Dane w formie wykresu	Tak	Tak, przez dodatek	Tak, przez dodatek
Przechowywanie danych w bazie cyklicznej	Tak	Tak, przez dodatek	Tak, przez dodatek
Przechowywanie danych w bazie relacyjnej	Nie	Tak, przez dodatek	Tak, przez dodatek
Powiadomienia o awarii	Nie	Tak, email lub telefon	Tak, email lub telefon
Wsparcie w lokalizacji awarii	Nie	Tak, poprzez mapę logiczną sieci	Tak, poprzez mapę logiczną sieci
Obsługa SNMP	Tak	Tak, przez wtyczkę	Tak, przez wtyczkę
Zbieranie danych spoza SNMP	Tak, niewielka liczba dostępnych metod	Tak, bogaty zestaw wtyczek	Tak, bogaty zestaw wtyczek
Monitorowanie pasywne	Nie	Tak	Tak
Nowoczesny interfejs użytkownika	Nie	Nie	Tak, z wykorzystaniem technologi AJAX
Wielu użytkowników	Tak	Tak	Tak
Metoda uwierzytelnienia	Uwierzytelnienie wewnętrzne	Uwierzytelnienie http	Uwierzytelnienie wewnętrzne
Zarządzanie kontami użytkowników z interfejsu	Tak	Nie	Tak
Definiowanie uprawnień dla użytkowników	Tak, przez interfejs graficzny	Nie	Tak, przez interfejs graficzny
Molarność	Nie	Nie	Tak
Rozmieszczenie modułów na różnych urządzeniach fizycznych	Nie dotyczy	Nie dotyczy	Tak

Kontynuacja na następnej stronie

Tablica 2.1 – Kontynuacja z poprzedniej strony

Nazwa systemu	Cacti	Nagios	Icinga
Możliwość monitorowania rozproszonego z instancją nadrzędną	Nie	Tak	Tak
Możliwość monitorowania rozproszonego bez instancji nadrzędnej	Nie	Tak, konieczny płatny dodatek	Tak
Generacja raportów	Nie	Nie	Tak, z wykorzystaniem JasperReports
Możliwość monitorowania klienta mobilnego	Nie	Nie	Nie
Dostępność	Darmowy	Częściowo darmowy, wiele płatnych elementów i funkcjonalności	Darmowy
Licencja	GPL v2	GPL v3 (tylko darmowe elementy)	GPL v2

Przedstawione systemy monitorujące w znacznym stopniu zaspokajają zapotrzebowanie rynku na systemy monitorowania. Pojawia się jednak pewna nisza związana z monitorowaniem urządzeń mobilnych. Zadanie to nie jest trywialne i wymaga obecności dodatkowych mechanizmów zarówno na urządzeniu mobilnym, jak i w innych elementach systemu. Żaden z analizowanych systemów nie posiadał w swej implementacji ani w oficjalnych repozytoriach z dodatkami, oprogramowania, które pozwalałoby na monitorowanie parametrów urządzenia mobilnego.

3. Monitorowanie klienta mobilnego

3.1. Monitorowanie rozproszone klientów statycznych

Firmy działające obecnie na rynku posiadają bardzo rozbudowaną infrastrukturę informatyczna. Od bardzo wielu lat działy odpowiedzialne za utrzymanie infrastruktury informatycznej prowadzą ciągły monitoring zarówno urządzeń sieciowych jak i serwerów oraz stacji roboczych użytkowników. Bardzo wiele firm posiada również specjalistyczne urządzenia, które również muszą być podłączone do sieci i monitorowane w celu zapewnienia ciągłości procesów biznesowych danej firmy. Wszystkie powyższe urządzenia rozumiane są jako klienty statyczne. Urządzenia tego typu zazwyczaj pracują nieprzerwanie lub w dobrze określonych przedziałach czasowych i posiadają dobrze zdefiniowaną hierarchię. Wzajemne relacje pomiędzy tymi urządzeniami wynikają w dużej mierze z struktury sieci lecz mogą również wynikać z roli jaką pełnią one w danej organizacji. Dzięki monitorowaniu wszystkich urządzeń w danej sieci systemy monitorujące są w stanie wspierać administratora wskazując z bardzo dużym prawdopodobieństwem miejsce wystąpienia awarii.

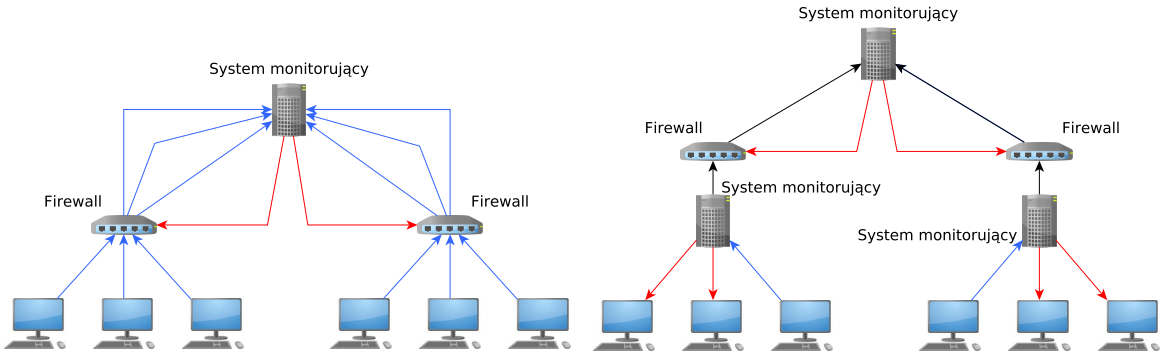
Sieć w dużej firmie rzadko stanowi jedną całość. Zazwyczaj są to segmenty sieci oddzielone zaporami lub w ogóle oddzielnie sieci LAN lub VLAN. Taka separacja urządzeń pozwala na zwiększenie poziomu bezpieczeństwa, lecz jednocześnie utrudnia monitorowanie całej infrastruktury. Aby umożliwić monitorowanie całej sieci firmowej wykorzystywane jest monitorowanie rozproszone. Można wyróżnić dwie podstawowe konfiguracje monitorowania rozproszonego:

Monitorowanie pasywne Istnieje jedna, centralna instancja jądra monitorującego, do którego przesyłane są wyniki sprawdzeń poszczególnych usług. Każde urządzenie samo monitoruje swoje usługi i zgłasza rezultaty.

Wieloinstancyjny system monitorujący Istnieje wiele instancji jądra monitorującego. Typowo, każda wydzielona część sieci posiada swoją instancję. Każda instancja może posiadać zarówno usługi monitorowane aktywnie jak i pasywnie. Wyniki sprawdzeń przesyłane są następnie do jednej wybranej instancji, która gromadzi wszystkie dane.

Wizualizację obu podstawowych konfiguracji przedstawiono na 3.1. Użycie monitorowania pasywnego dla wszystkich usług jest bardzo nie wygodnie i jednocześnie utrudnia konfiguracje, a także pozbawia administratora możliwości używania niektórych mechanizmów dostępnych wyłącznie dla urządzeń monitorowanych aktywnie. Ponadto wyniki sprawdzeń pasywnych nie są akumulowane, lecz wysyłane od razu po ich uzyskaniu. Oznacza to, że jeśli pojawi się chwilowy brak połączenia z serwerem, to wpisy dziennika zostaną zgubione. W przypadku, gdy jedynym celem systemu jest monitorowanie dostępności danej usługi zewnętrznej serwera, a nie jego parametrów wewnętrznych jest to jednak błąd pomijalny. Błąd ten staje

Rysunek 3.1. Monitoring pasywny oraz rozproszony. Kolor czerwony - monitorowanie aktywne, niebieski - monitorowanie pasywne, czarny - komunikacja wewnętrzna systemu.



się jednak istotny, gdy jednym z zadań systemu, jest gromadzenie i analiza danych historycznych. Wieloinstancyjny system monitorujący wymaga zdecydowanie więcej zasobów jednak pozwala na osiągnięcie znacznie wygodniejszego i bardziej niezawodnego systemu. Ponadto dzięki takiej konfiguracji nie ma potrzeby ingerencji w monitorowane serwery co redukuje ich obciążenie, a także zwiększa bezpieczeństwo. Warto również wspomnieć, że na przykład system Icinga, daje możliwość integracji wielu instancji jądra monitorującego, przy pomocy wspólnej bazy danych. Dzięki temu administrator danej sieci ma możliwość monitorowania i konfigurowania wielu instancji przy pomocy wspólnego interfejsu. Niestety w systemie Nagios rozwiązanie to zaliczane jest do części korporacyjnej tego systemu, przez co posiada zamknięte źródła i jego wykorzystanie wymaga zakupu licencji. Rozwiązania oparte na istnieniu jednej centralnej instancji jądra systemu monitorującego, są zazwyczaj darmowe. Niestety wymagają one dodatkowej instancji, zajmującej się agregacją danych. Należy również zwrócić uwagę, iż niektóre systemy jak Cacti nie posiadają w ogóle możliwości rozproszonego monitorowania.

3.2. Monitorowanie rozproszone klientów mobilnych

Rosnąca w ostatnich latach popularność technologii mobilnych przyczyniła się do pojawienia się w firmach bardzo dużej liczby urządzeń mobilnych, które wymagają zarówno zarządzania jak i monitorowania. Urządzenia mobilne są używane bardzo często przez przedstawicieli handlowych, a także przez menadżerów w celu umożliwienia wykonywania pracy poza obszarem firmy. Ponadto coraz więcej firm świadczących zaawansowane technicznie usługi wyposaża swoich pracowników w bardzo drogi sprzęt, który wymaga ciągłego monitorowania. Duże korporacje coraz częściej decydują się również na wyposażenie swoich pracowników w smartfony lub tablety, które mają ułatwić współpracę z firmą w trakcie podróży służbowych czy spotkań z klientami.

Klient mobilny posiada szereg cech, które znacząco odróżniają go od klientów statycznych. Przede wszystkim należy zauważyć, że urządzenia, o których mowa bardzo często pracują poza obszarem firmy. Wynika z tego, że nie zawsze możliwe jest utrzymywanie takich urządzeń w wirtualnej sieci prywatnej, gdyż urządzenie

może znaleźć się w obszarze, gdzie nie ma dostępu do internetu. Ponadto nie zawsze konieczne jest, aby urządzenia mobilne pracowały podłączone do sieci firmowej. Użytkownicy często wymagają jedynie dostępu do internetu i innych funkcji tego urządzenia. Warto więc zauważyć, że urządzenia te są często narażone na dostęp do sieci, o bardzo niskim poziomie zaufania i wielu zagrożeniach. Oznacza to w szczególności, iż urządzenie mobilne zazwyczaj posiada zmienny adres IP, który rzadko jest adresem globalnym. Również struktura sieci, z której korzystają klienty mobilne jest dynamiczna i znajduje się poza obszarem monitorowania administratorów danego przedsiębiorstwa. Znacząca większość klientów mobilnych dzięki kontaktom z siecią poza firmową posiada, w przeciwieństwie do klientów statycznych, możliwość synchronizacji swojego czasu czy to z serwerami czasu światowego, czy też z sieci GSM.

Należy również zwrócić uwagę na duże rozproszenie klientów mobilnych. W przeciwieństwie do klientów statycznych, którzy zazwyczaj pracują w pewnych grupach lub fragmentach sieci, klienty mobilne są zazwyczaj rozpatrywane pojedynczo. Większość klientów mobilnych operuje w pełni samodzielnie, zatem liczność grupy klientów wynosi 1. Powoduje to, że w przeciwieństwie do klientów statycznych gdzie grup koniecznych do wydzielenia było zazwyczaj kilka lub kilkanaście, w przypadku klientów mobilnych takich grup może być kilkaset lub nawet kilka tysięcy. Warto również dostrzec różnice w zasilaniu. Klienty mobilne zazwyczaj posiadają własne zasilanie, przez co każda operacja wykonywana na nim nie tylko spowalnia jego działanie, lecz również zmniejsza jego czas pracy pomiędzy ładowaniami. Przenośność klienta mobilnego zmienia również jego stopień bezpieczeństwa. Urządzenia mobilne stosunkowo często są gubione lub kradzione, co nie było możliwe w przypadku klientów statycznych. W związku z możliwością utraty urządzenia, nie powinno się na nim przechowywać tajnych danych, dzięki którym można by skompromitować cały system z którego korzysta klient.

Klient mobilny znacznie różni się swoją charakterystyką od klienta statycznego. Różni się również rodzaj monitorowanych usług. W przypadku klientów statycznych znaczna część wysiłków jest ukierunkowana na pomiar usług świadczonych przez dany system na rzecz innych systemów. Natomiast w przypadku klientów mobilnych istotniejsze wydaje się być monitorowanie parametrów wewnętrznych danego klienta.

3.3. Wymagania systemu monitorowania klientów mobilnych

Klient mobilny posiada zdecydowanie odmienną charakterystykę niż klient statyczny. Dokonano zatem analizy, jakie wymagania należy spełnić, aby dostarczyć system, który sprosta oczekiwaniom administratorów urządzeń mobilnych jak i statycznych.

Odbiorcą systemu mają być duże firmy i korporacje, które posiadają bardzo rozbudowaną sieć wewnątrz firmy, a ponadto udostępniają swoim pracownikom urządzenia mobilne różnej klasy. Wśród tych urządzeń znajdują się przede wszystkim telefony oraz tablety z systemem operacyjnym Android lub Windows Phone. Ponadto firma posiada także liczne laptopy wyposażone w system Windows lub Linux. Konieczne jest zatem, aby system pozwalał na monitorowanie każdej z wspomnianych platform. Duże firmy oraz korporacje, zazwyczaj posiadają już oprogramowanie służące do monitorowania swojej infrastruktury sieciowej. Aby umożliwić

administratorom łatwe zarządzanie oraz monitorowanie zarówno klientami mobilnymi jak i statycznymi, należy zapewnić integrację systemów monitorowania obu kategorii klientów. Dane odczytywane na urządzeniu mobilnym mogą zawierać zarówno dane prywatne pracownika, jak i tajemnice handlowe firmy. Oba te rodzaje danych należą do kategorii poufnych i powinny być należycie chronione. Ponieważ urządzenie mobilne będzie pracowało często poza siecią firmową, podczas tworzenia systemu należy zwrócić szczególną uwagę na kwestię bezpieczeństwa przesyłanych danych. System, musi przesyłać dane poprzez sieć publiczną, konieczne jest zatem zapewnienie odporności systemu na ataki zewnętrzne oraz na próby przekazywania sfałszowanych danych do systemu. Wszystkie wymagania stawiane przed omawianym systemem zostały zebrane w 3.1.

Tablica 3.1: Wymagania systemu monitorowania klienta mobilnego

Kod	Nazwa	Opis
W1	Spójność danych	System musi zapewnić, że wpisy dziennika nie zostaną zgubione. System musi zapewniać spójność danych pomiędzy serwerem, a klientem mobilnym.
W2	Integralności	System musi zapewnić, że wpisy dziennika dostarczone do serwera nie zostały w żaden sposób zmodyfikowane lub dodane.
W3	Autentyczność	System musi zapewnić, że odebrane dane pochodzą od uprawionego klienta.
W4	Poufność	System musi zapewniać poufność danych przesyłanych od klienta poprzez szyfrowanie.
W5	Dodawanie algorytmów	System musi być niezależny od algorytmu kryptograficznego stosowanego podczas przesyłania danych. Ponadto system musi umożliwiać dodawanie w prosty sposób nowych algorytmów kryptograficznych.
W6	Uwierzytelnienie klienta	System musi zapewnić możliwość uwierzytelnienia klienta.
W7	Wymienne algorytmy uwierzytelnienia klienta	System musi być niezależny od algorytmu uwierzytelnienia klienta. Ponadto system musi umożliwiać dodanie w prosty sposób nowych algorytmów uwierzytelnienia klienta.
W8	Uwierzytelnienie serwera	System musi zapewniać, iż wpisy dziennika zostaną przesłane tylko do wyznaczonego, uprawnionego serwera.
W9	Odporność na zgubienie urządzenia	System musi być odporny na zgubienie urządzenie. Oznacza to iż zgubienie urządzenia nie może powodować kompromitacji całego systemu.

Kontynuacja na następnej stronie

Tablica 3.1 – Kontynuacja z poprzedniej strony

Kod	Nazwa	Opis
W10	Dostarczanie w wiele miejsc	System musi umożliwiać przekazywanie danych do wielu podsystemów monitorujących, bez konieczności ich retransmisji z klienta mobilnego.
W11	Reguły definiowane dla każdego klienta	System musi umożliwiać definiowanie reguł dotyczących miejsc przeznaczenia dla każdego klienta indywidualnie.
W12	Oszczędność pasma	System powinien minimalizować ilość przesyłanych danych. Ponadto powinien skrócić do minimum czas oczekiwania na potwierdzenie przetworzenia przesłanych danych.
W13	Integracja z istniejącymi systemami	System monitoringu klienta mobilnego musi mieć możliwość integracji i współpracy z istniejącymi systemami monitorowania klienta statycznego.
W14	Analiza danych bieżących	System musi umożliwiać prezentację oraz analizę danych bieżących, a także posiadać możliwość reagowania na wystąpienie zdefiniowanych przez użytkownika zdarzeń.
W15	Analiza danych historycznych	System musi umożliwiać analizę zadanych danych historycznych włączając w to ich graficzną reprezentację.
W16	Kontrola danych wejściowych	System musi prowadzić kontrolę danych wejściowych od klientów. Konieczne jest aby system umożliwiał definiowanie jakie dane mogą być dostarczane przez jakich klientów.
W17	Łatwość dodawania nowych sprawdzeń	System musi umożliwiać dodawanie w łatwy sposób możliwości monitorowania nowych usług i parametrów.
W18	Klient dla platformy Android	System musi udostępniać klienta pozwalającego na monitorowanie urządzeń opartych na platformie Android
W19	Klient dla platformy Windows Phone	System musi udostępniać klienta pozwalającego na monitorowanie urządzeń opartych na platformie Windows Phone
W20	Klient dla platformy Windows 8	System musi udostępniać klienta pozwalającego na monitorowanie urządzeń opartych na platformie Windows 8
W21	Klient dla platformy Linux	System musi udostępniać klienta pozwalającego na monitorowanie urządzeń opartych na platformie Linux

3.4. Podstawowe decyzje projektowe

Przedstawione wymagania pozwalają na opracowanie systemu, który zaspokoi potrzebę monitorowania klienta mobilnego. System monitorujący stanowi duży zestaw programów wykonujących się na różnych urządzeniach i w różnych kontekstach. Zaprojektowanie i implementacja od podstaw systemu monitorującego, który spełniłby wszystkie przedstawione wymagania, daleko wykracza, poza ograniczenia czasowe pracy inżynierskiej. Ponadto dobre praktyki programistyczne, nakazują możliwie szerokie wykorzystanie gotowych programów. Należy również pamiętać, iż każdy program wymaga testowania i późniejszego utrzymania jego kodu. Wykorzystanie gotowego systemu, pozwala na uzyskanie niskim nakładem czasu systemu, który został już dokładnie przetestowany, a utrzymanie jego kodu zapewniane jest poprzez osoby zewnętrzne.

W związku z powyższym w niniejszej pracy podjęto decyzję, aby budowany system monitoringu klienta statycznego był oparty na jednym z dostępnych darmowych systemów monitorowania. Na podstawie analizy systemów monitorujących dostępnych na rynku, dokonanej w 2, został wybrany system monitorujący Icinga.

Wybór ten podyktowany jest wieloma zaletami tego systemu. Przede wszystkim, należy zauważyć przemyślaną architekturę. Posiada on budowę modularną, dzięki możliwe jest jego instalowanie na wielu odrębnych urządzeniach. Umożliwia to wzrost przepustowości całego systemu, a zatem pozwala na monitorowanie bardziej rozbudowanej sieci. Możliwe jest również monitorowanie infrastruktury w sposób rozproszony. System ten umożliwia zarówno monitorowanie pasywne, jak i wieloinstancyjne. Należy również wyróżnić system Icinga, gdyż jako jedyny udostępnia on w sposób darmowy, możliwość wspólnego zarządzania i podglądu wieloinstancyjnego systemu monitorującego. Nowoczesny i dynamiczny interfejs użytkownika dostarczany przez ten system, może być w łatwy sposób rozszerzany o dodatkowe funkcjonalności. Na szczególne uznanie zasługuje również rozbudowana i na bieżąco aktualizowana dokumentacja projektu. Najważniejszą z zalet jest jednak popularność tego systemu wśród administratorów. Dowodem popularności i wiarygodności systemu Icinga może być jego zastosowanie w ośrodku badań Europejskiej Organizacji Badań Jądrowych CERN[1].

Rdzeń monitorujący systemu Icinga jest przeznaczony dla systemu Linux, jednak możliwe jest uruchomienie go na większości systemów z rodziny Unix. W związku z powyższym wszelkie rozwiązania zaimplementowane w ramach tej pracy, są przeznaczone, dla tych samych systemów co jądro monitorujące.

[1] Szerszy opis zastosowania systemu Icinga w tej organizacji można znaleźć w [14].

4. System monitorowania Icinga

4.1. Opis systemu

System Icinga powstał jako klon systemu Nagios. Zachowana została kompatybilność wsteczna przez co możliwe jest używanie dodatków oraz wtyczek przeznaczonych dla systemu Nagios. Podstawowa konfiguracja systemu Icinga składa się dwóch modułów:

— Rdzeń monitorujący
— Interfejs użytkownika

Rdzeń monitorujący stanowi element centralny całego systemu. Do jego zadań należy przede wszystkim monitorowanie usług i urządzeń zgodnie z ustawieniami zawartymi w plikach konfiguracyjnych. System posiada możliwość monitorowania zarówno aktywnego jak i pasywnego. Monitorowanie aktywne wykonywane jest przy pomocy zewnętrznych programów lub skryptów nazywanych wtyczkami. Każda usługa oraz urządzenia posiada zdefiniowaną komendę sprawdzającą która definiuje jaką wtyczkę należy uruchomić oraz jakie dane do niej przekazać. Każda wtyczka posiada zdefiniowany zestaw danych wejściowych, które odpowiadają jej opcjom konfiguracyjnym. Po zakończeniu pomiaru wtyczka przekazuje dane o jego wynikach do systemu monitorującego. Przekazanie to odbywa się dwiema drogami. Pierwsza z nich to wartość zwrócona z programu, która determinuje w jakim stanie znajduje się urządzenie lub usługa. Zwrócona wartość powinna być jedną z następujących:

0 OK, wtyczka mogła wykonać sprawdzenie i usługa lub urządzenie jest w stanie OK
1 WARNING, wtyczka mogła wykonać sprawdzenie ale parametry urządzenia lub usługi przekraczają poziom ostrzegawczy.
2 CRITICAL, wtyczka mogła wykonać sprawdzenia ale parametry urządzenia lub usługi przekraczają poziom krytyczny.
3 UNKNOWN, wtyczka nie była w stanie wykonać sprawdzenia ze względu na dostarczenie nie prawidłowych parametrów wywołania lub niskopoziomowego błędu systemu.

Większość wtyczek dokonuje pewnych pomiarów, dlatego każde ich wykonanie gromadzi zdecydowanie więcej danych niż można przekazać poprzez jedną z czterech wartości. Przekazanie pozostałych informacji odbywa się poprzez dane tekstowe, wypisywane przez wtyczkę na standardowym wyjściu programu. Dane te rozdzielane są znakiem | na dwie grupy. Przed tym znakiem, znajdują się dane czytelna dla człowieka. Po znaku znajdują się dane wydajnościowe w formacie klucz=wartość, przeznaczone do analizy przez zewnętrzne programy np. do generacji wykresów. Znak | oraz druga grupa nie są obowiązkowe.

System Icinga umożliwia również monitorowanie w sposób pasywny. Dostarczanie wyników sprawdzenia pasywnego odbywa się poprzez plik komend zewnętrznych. Plik komend zewnętrznych jest to potok nazwany przez który poszczególne komendy trafiają do rdzenia monitorującego. Pełna lista dostępnych komend znajduje się w [5, 412-436]. Zatem pewien dowolny program wykonuje sprawdzenia urządzenia lub usługi, po czym zapisuje wynik tego sprawdzenia zgodnie z formatem do potoku. Format przekazywanego rezultatu sprawdzenia został opisany w [5, 296-299]. Jeśli program wykonuje się na urządzeniu innym niż to na którym uruchomiony jest rdzeń monitorujący konieczne jest przesłanie danych do tego systemu. System Icinga posiada do tego celu dodatek NSCA, który został szeroko opisany w 4.4

Dane otrzymane od wtyczki są przez system monitorujący przechowywane wraz z innymi danymi potrzebnymi do monitorowania w plikach, a gdy przestają one być potrzebne, są kasowane. System Icinga udostępnia jednak możliwość przechowywania tych danych w bazie danych przy pomocy komponentu systemu IDOUtils. Został on dokładnie opisany w 4.2. Rdzeń monitorujący posiada także możliwość udostępniania danych wydajnościowych otrzymanych przez każdą wtyczkę dla zewnętrznych programów. Możliwe jest zdefiniowanie formatu danych wyjściowych. Po włączeniu eksportu danych wydajnościowych, rdzeń monitorujący będzie zapisywał do zdefiniowanych plików dane wydajnościowe pochodzące od wtyczek jak i czasy ich przybycia. Dane te są wykorzystywane np. przez dodatek inGraph opisany w 4.3.

System Icinga odziedziczył po systemie Nagios klasyczny interfejs oparty na technologii CGI. Ponieważ technologia ta jest już przestarzała, udostępniono użytkownikom również w pełni nowoczesny interfejs nazywany icinga-web. Jest to dynamiczny interfejs użytkownika zaimplementowany w języku PHP przy wykorzystaniu technologii AJAX. Różnice pomiędzy tymi interfejsami zauważalne są nie tylko w warstwie prezentacji lecz również w architekturze i warstwie komunikacji z rdzeniem monitorującym. Interfejs klasyczny wykorzystywał pliki generowane przez rdzeń monitorujący do pobierania aktualnych danych, przez co musiał on znajdować się na tym samym urządzeniu co rdzeń monitorujący. W przypadku interfejsu icinga-web komunikacja z rdzeniem monitorującym odbywa się poprzez bazę danych, dzięki czemu elementy te mogą znajdować się na różnych fizycznych urządzeniach. Ponadto interfejs ten wykorzystuje drugą bazę danych o bardzo prostym schemacie jako miejsce przechowywania danych konfiguracyjnych. Należy również dostrzec różnice w bezpieczeństwie. Interfejs klasyczny zabezpieczony był jedynie poprzez mechanizm uwierzytelnienia http i wszyscy użytkownicy mieli dostęp do całego systemu. Interfejs icinga-web posiada swoją bazę użytkowników oraz ich uprawnień. Umożliwia to ograniczenie praw danego użytkownika np. tylko do wyświetlania stanu konkretnego urządzenia lub usługi.

4.2. Komponent IDOUtils

Komponent IDOUtils jest to zestaw programów dzięki którym możliwe jest składowanie informacji generowanych przez rdzeń monitorujący w bazie danych. W wersji dostępnej podczas pisania tej pracy wspierany były następujące systemy zarządzania bazą danych:

— MySql,

— PostgreSQL,
— Oracle.

W celu zapewnienia funkcjonalności omawianego komponentu, konieczne jest utworzenie bazy danych o odpowiednim schemacie, który został opisany w [5, 669-750]. Udostępnione zostały również skrypty SQL, które definiują odpowiednie tabele. Ponadto administrator musi zapewnić odpowiednia konfigurację bazy danych, w tym konto użytkownika i hasło, w taki sposób, aby umożliwić odpowiednim elementom komponentu IDOUtils dostęp do bazy danych.

W celu odciążenia urządzenia, na którym uruchomiony jest system Icinga. Komponent ten został podzielony, na kilka elementów, które mogą znajdować się na różnych urządzeniach. Można wyróżnić następujące elementy:

IDOMOD moduł rdzenia monitorującego, który pozwala mu na dostęp do bazy danych

LOG2IDO program pozwalający na import utworzonych wcześniej plików do bazy danych

FILE2SOCK program pozwalający na przekierowanie danych zapisywanych do pliku do gniazda TCP lub Unix

IDO2DB demon, który jest odpowiedzialny za wykonywanie operacji na bazie danych

Podstawowymi elementami całego komponentu są IDOMOD oraz IDO2DB. Moduł rdzenia IDOMOD ładowany jest przez rdzeń systemu Icinga tuż po starcie. Po załadowaniu zapewnia on spójny interfejs do uzyskiwania danych dla wszystkich pozostałych części rdzenia monitorującego. Ponieważ wykonywanie operacji na bazie danych może być czasochłonne nie powinno być to wykonywane przez rdzeń monitorujący. Z tego powodu powstał program IDO2DB. Jest on uruchomiony jako demon na dowolnym urządzeniu. Zadaniem tego serwisu jest fizyczna realizacja żądań na bazie danych.

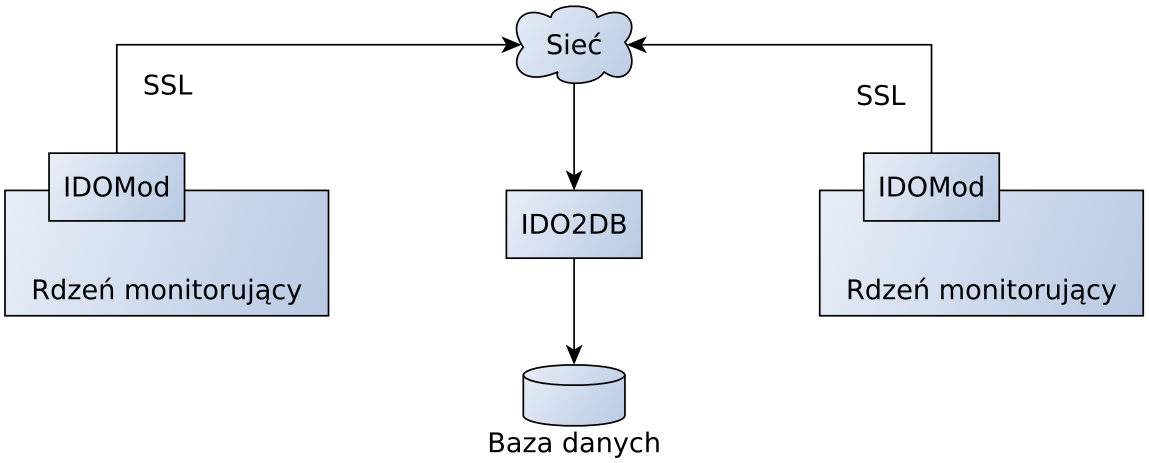
Ponieważ rdzeń monitorujący oraz demon IDO2DB mogą znajdować się zarówno na jednym urządzeniu jak i na różnych urządzeniach konieczne jest zapewnienie odpowiednich mechanizmów komunikacji pomiędzy nimi. Gdy programy te znajdują się na różnych urządzeniach, jako mechanizm komunikacji wykorzystywane są gniazda TCP. W podstawowej konfiguracji dane przekazywane są w sposób niezaszyfrowany. Jeśli jednak istnieje potrzeba zapewnienia tajności oraz integralności przekazywanych danych możliwe jest użycie protokołu SSL[1]. W sytuacji, gdy oba programy uruchomione są na tym samym urządzeniu, w celu poprawy wydajności możliwe jest użycie gniazd protokołu Unix[2]. Najpopularniejszy sposób użycia został przedstawiony schematycznie na 4.1.

W celu zapewnienia możliwości migracji z środowiska, które korzystało wcześniej z przechowywania danych w plikach, został dostarczony program LOG2IDO. Pozwala on, na import danych historycznych do bazy danych. Program ten, analogicznie jak IDOMOD nie operuje bezpośrednio na bazie danych, lecz komunikuje się tymi samymi metodami co IDOMOD z demonem IDO2DB. Zarówno program LOG2IDO jak i moduł IDOMOD mogą kierować żądania do IDO2DB poprzez plik.

[1] ang. *Secure Socket Layer* – protokół warstwy prezentacji, zapewniający poufność oraz integralność przesyłanych danych.

[2] and. *Unix Domain Socket* – metoda komunikacji między procesowej w systemach Unix. Posiada jednolite API jak gniazda domeny internetowej.

Rysunek 4.1. Schemat wykorzystania IDOUtils w systemie Icinga.



W celu zapewnienia przekazywania tych danych z pliku do demona IDO2DB opracowano program FILE2SOCK. Jest to prosty program, który przekazuje dane zapisane do danego pliku do demona IDO2DB. Program ten nie zajmuje się w żadnym stopniu przetwarzaniem odczytaniem danych, lecz jedynie przesłaniem ich poprzez gniazdo internetowe lub Unix do demona IDO2DB.

4.3. Dodatek inGraph

inGraph jest to dodatek do systemów Icinga oraz Nagios, który umożliwia prezentację danych zgromadzonych poprzez system monitorujący w postaci wykresów. Dodatek ten został opracowany przez firmę NETWAYS GmbH i wydany na licencji GPL w wersji 3. Cechą, która odróżnia dodatek inGraph od innych rozwiązań, przeznaczonych do analizy danych historycznych jest wykorzystanie relacyjnej bazy danych do przechowywania danych otrzymanych od systemu monitorującego. Dane, Na podstawie otrzymanych danych dodatek inGraph dokonuje przeliczeń dla odpowiednich przedziałów czasowych, które używane są do późniejszej generacji wykresów. Rozmiary przedziałów definiowane są przez użytkownika w plikach konfiguracyjnych. Wykorzystanie tego rodzaju bazy danych powoduje nieustanny wzrost rozmiaru bazy. W celu optymalizacji zajętości przestrzeni dyskowej dodatek inGraph administruje danymi zgodnie z polityką zdefiniowaną w plikach konfiguracyjnych. Dla każdego przedziału czasowego zdefiniowany jest również okres przechowywania danych. Dodatek umożliwia przeglądanie danych dokładnych z najmniejszych przedziałów czasu, jak i wykresów długoterminowych prezentujących trendy danej wartości. Ponieważ dane są bezpośrednio administrowane przez dodatek inGraph, możliwa jest zmiana czasów przechowywania danych z wskazanych przedziałów nawet w trakcie działania systemu[3].

Dodatek inGraph składa się z dwóch niezależnych elementów, komunikujących się poprzez XML-RPC[4]:

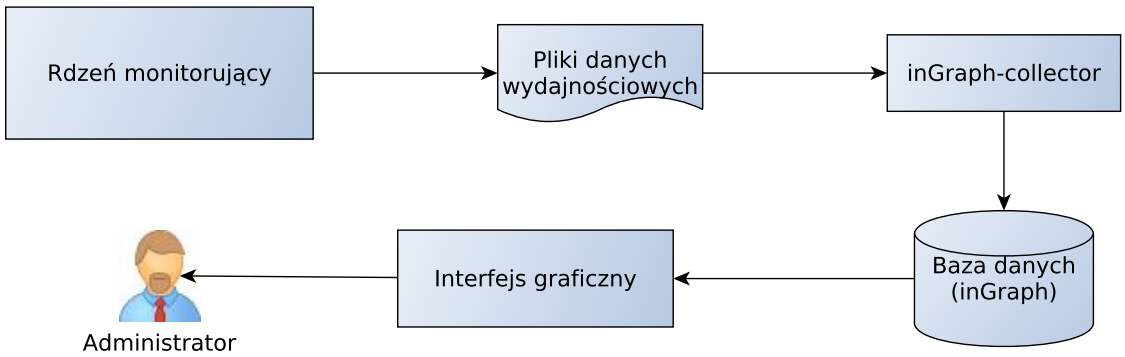
[3] Dla porównania należy przypomnieć systemy oparte na cyklicznych baza danych, gdzie rozmiar definiowany może być tylko i wyłącznie podczas tworzenia bazy.

[4] ang. *XML Remote Procedure Call* – zdalne wywołanie procedur przy użyciu XML. Metoda zdalnego wywoływania funkcji oparta na dokumentach w formacie XML. Szczegółowy opis w [11].

— interfejs graficzny,
— rdzeń zbierający dane.

Rdzeń zbierający oraz przetwarzający dane został napisany w języku Python. Jego zadaniem jest pobieranie danych od systemu monitorującego, dokonywanie ich przeliczeń, oraz umieszczanie ich wyników w bazie danych. Do pobierania danych z systemu monitorującego wykorzystano mechanizm udostępniania danych wydajnościowych. System monitorujący, musi eksportować dane przy pomocy formatu zrozumiałego dla dodatku inGraph. Demon zbierający dane dokonuje analizy otrzymanych danych, a następnie wykonuje wszystkie niezbędne obliczenia, a wyniki zapisuje w bazie danych MySQL lub PostgreSQL. Ważną różnicą pomiędzy danymi składowanymi w tej bazie, a danymi przechowywanymi przez system monitorujący jest ich format. Systemy monitorujące, przechowują w postaci numerycznej jedynie skwantowany stan danej usługi lub urządzenia. Dodatek inGraph przechowuje natomiast w swojej bazie dane w postaci już przetworzonej. Oznacza to iż dokonywany jest rozbiór składniowy rezultatów pomiarów i w bazie danych zapamiętywane są pochodzące z tych rezultatów dane w postaci numerycznej. Typowy przepływ danych został przedstawiony na 4.2.

Rysunek 4.2. Typowy przepływ danych przy wykorzystaniu dodatku inGraph.

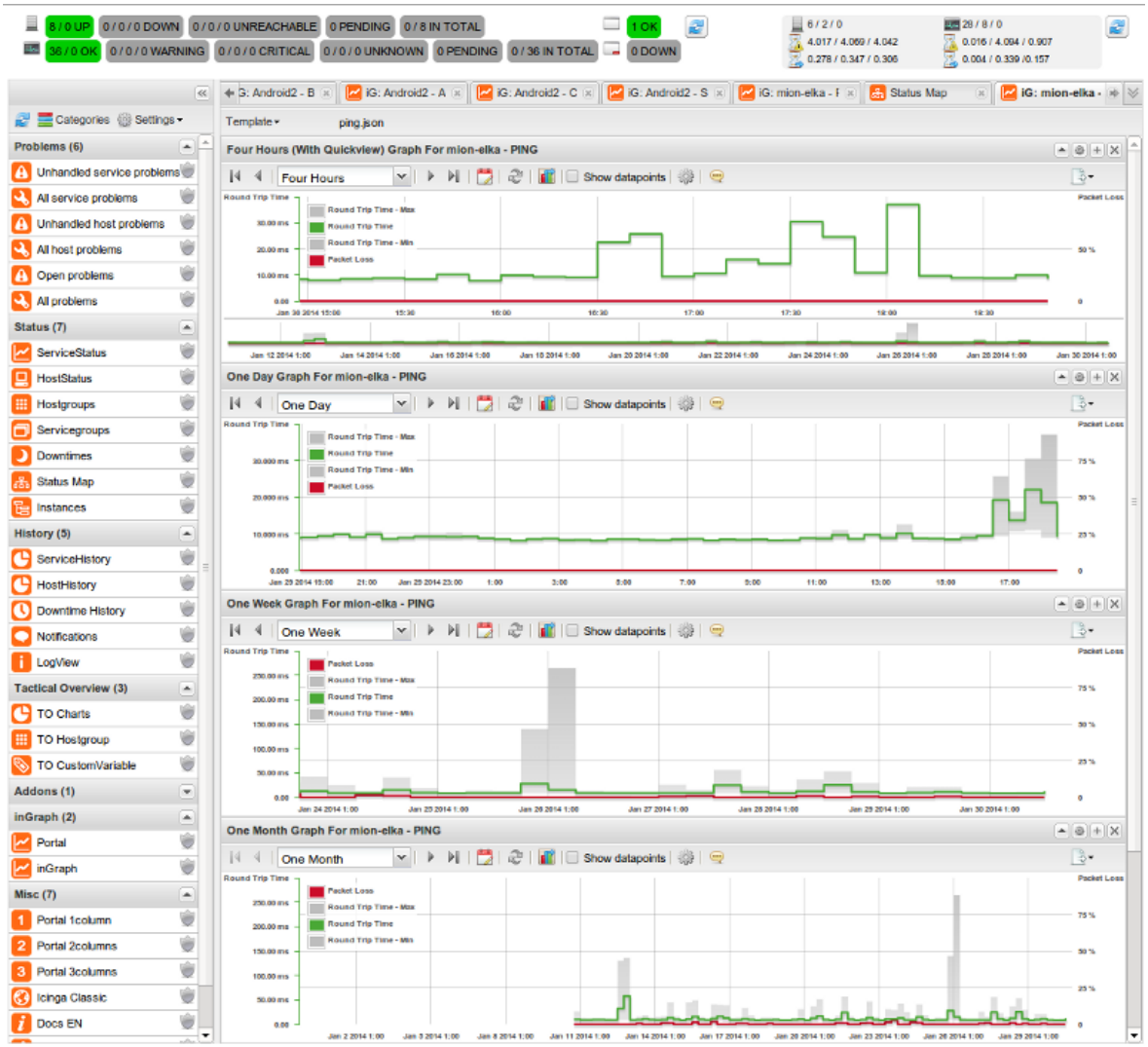


Interfejs użytkownika został napisany w językach PHP oraz JavaScript. Umożliwia on podgląd danych zebranych i przetworzonych przez rdzeń dodatku. Interfejs może funkcjonować zarówno jako niezależny serwis jak i jako integralna część interfejsu systemu Icinga. Umożliwia on generację wykresów dla każdego z urządzeń oraz dla każdej z usług. Formaty wykresów, a także przedziały agregacji danych definiowane są w plikach konfiguracyjnych w formacie JSON[5]. Użytkownik po wybraniu usługi lub urządzenia uzyskuje interaktywny wykres przezentujący dane w zadanym okresie. Wszystkie wykresy wygenerowane przez program są w pełni konfigurowalne jak i edytowalne. Typ prezentowanych danych jest uzależniony od rozmiaru przedziału czasu w którym generowany jest wykres. Jeśli okno czasu jest odpowiednio małe, na wykresie zostaną przedstawione dane dokładne. W sytuacji, gdy nie jest możliwe przedstawienie danych dokładnych, ze względu na rozmiar zadanego okresu czasu, dane są agregowane w przedziały, a na wykresie udostępniana jest wartość minimalna, maksymalna oraz średnia dla danego przedziału

[5] ang. *JavaScrip Object Notation* – lekki format tekstowy wymiany danych komputerowych. Szczegółowo opisany w [6].

agregacji danych. Przykładowe wykresy wygenerowany przy pomocy dodatku inGraph można zobaczyć na 4.3. Szczególną uwagę warto zwrócić na szare pola reprezentujące minimum oraz maksimum w danym przedziale.

Rysunek 4.3. Interfejs dodatku inGraph.



4.4. Dodatek NSCA

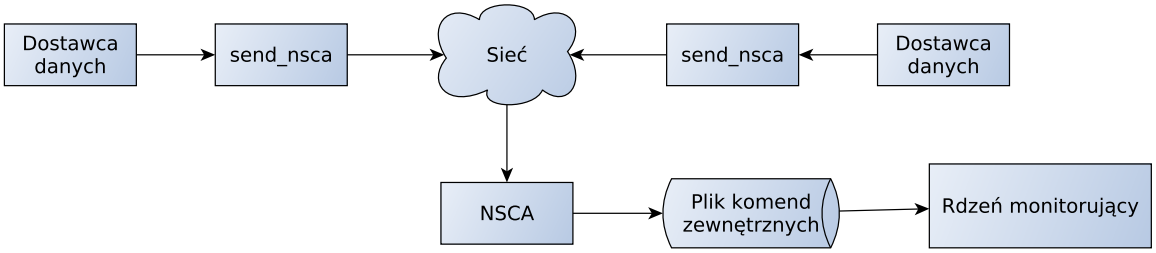
4.4.1. Opis dodatku NSCA

NSCA - Nagios Service Check Acceptor jest to dodatek do systemów monitorujących opartych o system Nagios, więc również systemu Icinga. Pozwala on na wykorzystanie mechanizmów pasywnego monitorowania z systemu innego niż ten na którym uruchomione jest oprogramowanie monitorujące. Program ten został napisany w całości w języku C i wydany na licencji pozwalającej na wgląd do kodu źródłowego. Wykorzystuje on plik zewnętrznych komend i nie integruje się z rdzeniem monitorującym. Dzięki temu możliwe jest jego wykorzystanie zarówno

w systemie Nagios jak i jego klonach takich jak system Icinga. Dodatek ten składa się z dwóch modułów:

— moduł wysyłający (send_nsca) służący do wysyłania wyników sprawdzeń z monitorującego systemu do centralnego serwera, na którym umieszczony jest rdzeń systemu monitorującego odpowiedzialny za przetwarzanie wyników sprawdzeń,
— moduł odbierający (nsca) służący do odbierania wyników sprawdzeń od klientów i dostarczaniu ich do pliku komend zewnętrznych danego systemu monitorującego.

Rysunek 4.4. Schemat działania dodatku NSCA.



Moduł wysyłający

Ta część dodatku uruchamiana jest na systemie, na którym funkcjonuje jakiś mechanizm sprawdzający, który generuje wpisy dziennika. Wpisy te po utworzeniu, przekazywane są do programu wysyłającego. Moduł wysyłający, po uruchomieniu odczytuje ustawienia z pliku konfiguracyjnego, a następnie próbuje połączyć się z serwerem. Po udanej próbie połączenia otrzymuje pakiet inicjujący, który zawiera:

wektor inicjujący używany do celów kryptograficznych, wygenerowany przez serwer pseudolosowy ciąg znaków, konieczny do inicjalizacji algorytmu kryptograficznego,
stempel czasu czas odczytany przez serwer w chwili nadejścia połączenia od klienta.

Po otrzymaniu pakietu inicjującego moduł rozpoczyna czytanie wpisów z standardowego wejścia programu. Wszystkie wpisy dziennika muszą być odpowiednio sformatowane. Poszczególne pola informacyjne muszą być rozdzielone pojedynczą tabulacją, a cały wpis zakończony znakiem nowej linii. Wpisy dotyczącego urządzenia powinny zawierać następujące pola:

nazwa urządzenia krótka nazwa urządzenia, którego stan jest przekazywany,
stan numerycznie wyrażony kod stanu urządzenia,
odczyt dodatkowe wartości odczytów opisujące stan urządzenia.

Natomiast wpisy dotyczące usługi świadczonej przez to urządzenia, lub innego rejestrowanego parametru tego urządzenia powinny zawierać następujące pola:

nazwa urządzenia krótka nazwa urządzenia na którym uruchomiona jest usługa,
opis usługi nazwa usługi danego urządzenia, której dotyczy wpis
stan numerycznie wyrażony kod stanu usługi,
odczyt dodatkowe wartości odczytów opisujące stan usługi.

Łatwo zauważyć, że żadne z pól wpisu dziennika nie zawiera stempla czasu wymaganego przez rdzeń sprawdzający przy zapamiętywaniu odczytu pasywnego. Dzieje się tak, gdyż program NSCA posiada zdefiniowaną własną politykę określania czasu wpisu w dzienniku. Do każdego pakietu zawierającego wpis dziennika dodawany jest stempel czasu otrzymany w pakiecie inicjującym od modułu odbierającego. Właściwy stempel czasu, który trafia do jadra sprawdzającego nadawany jest natomiast przez moduł odbierający.

Kolejnym krokiem działania modułu jest obliczenie cyklicznego kodu nadmiarowego CRC32 dla danego pakietu. Po dołączeniu obliczonego kodu do pakietu pakiet jest szyfrowany. Algorytm kryptograficzny stosowany do szyfrowania pakietów został wcześniej zainicjalizowany wektorem pseudolosowych danych odebranych w pakiecie inicjalizacyjnym od modułu odbierającego. Po zaszyfrowaniu dane są wysyłane, a moduł wysyłający, bez oczekiwania na potwierdzenie przetworzenia przez serwer, rozpoczyna przetwarzanie kolejnego wpisu dziennika.

Moduł odbierający

Demon, który stanowi moduł odbierający funkcjonuje na tym samym systemie operacyjny na którym znajduje się rdzeń systemu monitorującego. Ta część odpowiedzialna jest za odbieranie danych od klientów i przekazywanie ich do rdzenia programu monitorującego. Moduł ten może pracować w jednym z poniższych trybów:

samodzielny demon jedno procesowy uruchomiony w tle demon, który nasłuchuje na przychodzące połączenia od klientów i po nadejściu połączenia jest ono obsługiwane przy użyciu jednego procesu z jednym wątkiem,

samodzielny demon wieloprocesowy uruchomiony w tle demon, którego proces główny nasłuchuje na nadejście połączeń od klientów, gdy takie połączenie nadejdzie proces jest duplikowany i każdy z klientów obsługiwany jest w innym procesie potomnym,

demon zintegrowany z inetd w systemie uruchomiony jest demon inetd, który nasłuchuje na połączenia od klientów na konkretnym gnieździe, a gdy nadejdzie połączenie od klienta uruchamiany jest proces demona NSCA, który obsługuje nowe połączenie i kończy się wraz z zakończeniem obsługi klienta

Do przekazywania wpisów dziennika używany jest mechanizm pasywnego monitorowania dostępny w systemach z rodziny Nagios. Aby możliwe było wykorzystanie tego mechanizmu konieczne jest zapewnienie demonowi dostępu do pliku zewnętrznych komend systemu monitorującego. Ponieważ plik zewnętrznych komend jest potokiem nazwanym, chroniony jest on przez Uniksowy system uprawnień użytkowników. Zapewnienie dostępu do takiego bytu może się odbyć na dwa sposoby. Pierwszym, polecanym przez twórców systemów monitorujących, jest uruchamianie demona NSCA jako procesu tego samego użytkownika co proces rdzenia systemu monitorującego. Drugim sposobem jest modyfikacja praw dostępu do omawianego pliku, tak aby umożliwić dostęp użytkownikowi z którego uprawnieniami uruchomiony jest demon NSCA. Przy zastosowaniu drugiego rozwiązania zalecana jest szczególna ostrożność, gdyż dostęp do pliku zewnętrznych komend daje bardzo duże możliwości ingerencji w system monitorujący.

Komunikacja modułu odbierającego z klientem rozpoczyna się od nadejścia połączenia od klienta. Gdy moduł odbierający otrzyma nowe połączenie zostanie wysłany pakiet inicjalizujący, którego zawartość została opisana w 4.4.1. Po przesłaniu pakietu inicjalizującego połączenie, moduł odbierający oczekuje na dane od klienta. Każdy wpis dziennika przesyłany jest przy użyciu pakietu o poniższych polach:

wersja protokołu aktualnie używana wersja protokołu komunikacyjnego,
kod CRC32 kod CRC32 bieżącego pakietu,
stempel czasu: stempel czasu pochodzący z pakietu inicjalizującego przesłanego klientowi,
kod statusu kod stanu usługi/hosta powiązany z przesyłanym wpisem
nazwa hosta: nazwa klienta, który podlegał sprawdzeniu. Nie jest konieczne aby był to ten sam klient, który dostarcza dane,
opis usługi nazwa usługi, która podlegała sprawdzeniu lub pusty napis jeśli sprawdzenie dotyczy hosta,
wynik sprawdzenia napis wygenerowany przez wtyczkę, która dokonywała sprawdzenia, zawierający dodatkowe dane na temat stanu urządzenia lub usługi

Pakiety są zaszyfrowane z użyciem algorytmu oraz klucza symetrycznego pochodzącego z pliku konfiguracyjnego. Po odebraniu spodziewanej ilości danych, następuje próba odszyfrowania odebranych danych. Sprawdzenie poprawności odebranych danych i jednocześnie weryfikacja uprawnień odbywa się poprzez kontrolę zawartości pola CRC32. Jeśli wartość znajdująca się w tym polu, zgadza się z wartością wyliczoną dla całości otrzymanych danych, to pakiet jest przyjmowany, w przeciwnym zaś razie pakiet zostanie odrzucany. Dalsze przetwarzanie otrzymanego pakietu rozpoczyna się od porównania bieżącego stempla czasu z tym pochodzącym z odebranego pakietu. Jeśli różnica pomiędzy nimi jest zbyt duża, dane zostają odrzucone. Ostatnią czynnością wykonywaną przez moduł odbierający jest zapisanie odebranego wpisu do pliku zewnętrznych komend jądra systemu monitorującego.

Warto wspomnieć, że stempel czasu przesłany przez klienta nie jest dostarczany do jądra monitorującego. Służy on jedynie określeniu odstępu czasu od inicjalizacji sesji do chwili otrzymania wiadomości i podjęciu decyzji o przyjęciu, bądź odrzuceniu pakietu. Do systemu monitorującego trafia natomiast bieżący stempel czasu serwera, na którym uruchomiony jest moduł odbierający i jądro systemu monitorującego. Do generacji stempla czasu wykorzystywany jest czas uniwersalny. Istotną, może się również okazać informacja, iż protokół komunikacyjny nie przewiduje przesyłania ACK[6], bądź też NACK[7]. Moduł wysyłający, ma zatem pewność, iż wysłane przez niego dane zostaną dostarczone, gdyż używany jest protokół TCP. Nie ma jednak żadnej gwarancji ani informacji, że dane przesłane do modułu odbierającego zostaną dostarczone do rdzenia systemu monitorującego.

[6] ang. *Acknowledgement* – pozytywne potwierdzenie, powszechnie przyjęta nazwa komunikatu potwierdzającego przyjęcie i przetworzenie danych przez aplikację

[7] ang. *Negative Acknowledgement* – potwierdzenie negatywne, powszechnie przyjęta nazwa komunikatu oznaczająca odmowę przyjęcia lub przetworzenia odebranych danych

4.4.2. Bezpieczeństwo

Bezpieczeństwo monitorowania z użyciem dodatku NSCA opiera się na kryptografii symetrycznej oraz cyklicznym kodzie nadmiarowym CRC32. Wiadomość inicjująca połączenie jest nieszyfrowana. Natomiast każda wiadomość zawierająca wpisy dziennika jest zaszyfrowana algorytmem wybranym podczas konfiguracji systemu. Dodatek NSCA korzysta z biblioteki libmcrypt[8] i umożliwia użycie jednego spośród wielu algorytmów kryptografii symetrycznej, które zostały w niej zaimplementowane. Użytkownik posiada jedynie możliwość wyboru stosowanego algorytmu, natomiast jako tryb pracy stosowany jest tryb sprzężenia zwrotnego szyfrogramu. Tryb ten wymaga zawsze inicjalizacji zarówno kodera jak i dekodera tym samym wektorem początkowym, który w przypadku tego protokołu, jest przesyłany przez serwer w pakiecie inicjującym.

Wszystkie algorytmy symetryczne do prawidłowego działania wymagają, aby komunikujące się strony współdzieliły pewien sekret jakim jest klucz używany do szyfrowania. Ujawnienie klucza symetrycznego wiąże się z kompromitacją całego systemu kryptograficznego. W dodatku NSCA klucz ten uzyskiwany jest z hasła, które musi być zapisane przez administratora systemu zarówno w części odbierającej jak i wysyłającej. Oczywistym jest, iż poza współdzieleniem klucza, wszystkie komunikujące się węzły muszą używać tego samego algorytmu kryptograficznego.

Algorytmy szyfrowania zapewniają tajność przesyłanej wiadomości, jednak w przypadku systemu monitorowania potrzebne jest również zapewnienie integralności wiadomości. Integralność w dodatku NSCA zapewniana jest poprzez cykliczny kod nadmiarowy CRC32. Obliczanie kodu CRC32 odbywa się poprzez dzielenie przesyłanego ciągu bitów przez dzielnik o długości 33 bitów, co daje kod CRC o długości 32 bitów. W celu sprawdzenia integralności, otrzymane bity są dzielone przez kod CRC. Jeśli reszta z dzielenia jest zero, oznacza to poprawna weryfikację integralności wiadomości. Jeśli reszta z dzielenia jest niezerowa oznacza to naruszenie integralności przesłanej wiadomości. W szczególności, taka sytuacja może się zdarzyć, gdy klient używa innego algorytmu kryptograficznego lub klucza. Pakiety, których integralność nie zostanie pozytywnie zweryfikowana są odrzucane.

Model bezpieczeństwa zastosowany w dodatku NSCA ma wiele wad. Największą z nich jest zastosowanie kodu CRC32 do sprawdzania integralności przesyłanych wiadomości. Kod ten można bardzo prosto i szybko obliczyć, a ponadto posiada on niewielką długość. Niestety jest on podatny na kolizje przez co nie powinien on być stosowany w kryptografii. Problem znalezienia kolizji kodu CRC można sprowadzić w łatwy sposób do problemu wspólnych urodzin opisanego w [2]. Prawdopodobieństwo nie znalezienia żadnej kolizji po obliczeniu 200 000 kodów wynosi poniżej 1%. Prawdopodobieństwo nie znalezienia kolizji w zależności od liczby obliczonych kodów CRC32 przedstawiono w 4.1. Łatwość odnalezienia kolizji nie jest jedyną wadą modelu bezpieczeństwa zastosowanego w dodatku NSCA. Warto przypomnieć, iż wszystkie ustawienia zarówno modułu wysyłającego jak i odbierającego przechowywane są w plikach na dyskach odpowiednich urządzeń. Pliki te zawierają również klucze symetryczne, które są stosowane w całym systemie. Oznacza to, iż uzyskanie dostępu typu odczyt do takiego pliku powoduje utratę tajności danych przesyłanych w całym systemie. Ponadto przyjęty model bezpieczeństwa, nie zawiera żadnej weryfikacji danych pochodzących od klientów. Oznacza to, że każdy klient może przesłać wpisy dziennika, udające wpisy pochodzące od zupełnie

[8] Szczegółowy opis biblioteki jak i dostępnych w niej algorytmów można znaleźć w [7].

Liczba obliczeń	Prawdopodobieństwo
50 000	74,7%
77 000	50,1%
78 000	49,2%
102 000	29,8%
110 000	24,5%
128 000	14,8%
150 000	7,3%
200 000	0,95%

Tablica 4.1: Prawdopodobieństwo nie znalezienia kolizji w zależności od liczby obliczonych kodów CRC32

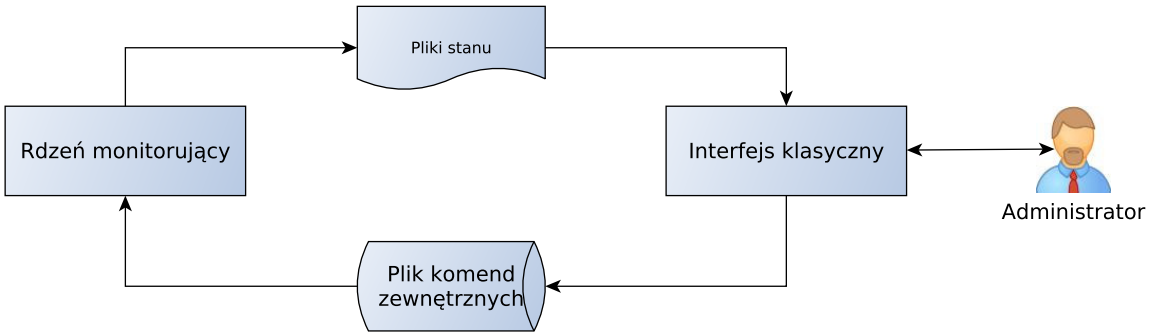
innych klientów. W szczególności jeśli atakujący uzyska klucz symetryczny, to nie tylko będzie mógł odczytywać informacje o wpisach przesyłanych od klientów, lecz także podszywać się pod klientów i przesyłać fałszywe wpisy. Taka luka może być wykorzystana przy ataku na jakąś usługę lub urządzenie. Atakujący rozpoczyna atak, po czym przechwytuje pakiety z wpisami dziennika, które mogą świadczyć o rozpoczęciu ataku i w zamian przesyła do serwera fałszywe pakiety informujące, iż wszystkie usługi pracują normalnie.

4.5. Podstawowe konfiguracje rozproszone

Podstawowa konfiguracja systemu monitorującego Icinga składa się jedynie z rdzenia monitorującego oraz klasycznego interfejsu użytkownika. W tej konfiguracji, zarówno ustawienia systemu monitorującego, jak i dane o stanie usług i urządzeń znajdują się w plikach lokalnych. Jeśli nie zostaną użyte żadne dodatkowe mechanizmy transportu danych, obie części systemu Icinga będą musiały być wykonywane na jednym urządzeniu. Jeśli monitorowana infrastruktura jest bardzo rozbudowana, a administrator często i intensywnie korzysta z interfejsu graficznego, to umiejscowienie obu tych elementów na jednym urządzeniu może powodować jego znaczące obciążenie i zaburzenia w prawidłowym monitorowaniu infrastruktury. Należy również zwrócić uwagę na zagadnienie bezpieczeństwa takiego rozwiązania. Jeśli administrator chciałby udostępnić interfejs użytkownika poza monitorowaną sieć, musi on zezwolić na dostęp z zewnątrz do urządzenia, które monitoruje całą infrastrukturę. Obniża to bezpieczeństwo w sieci, gdyż atakujący może ukierunkować swoje działania właśnie na to urządzenie, a uzyskanie dostępu do niego pozwoli na ataki innych, być może słabiej zabezpieczonych urządzeń znajdujących się w sieci. Schemat opisanej konfiguracji przedstawiono na 4.5.

Podstawową metodą optymalizacji przedstawionej konfiguracji jest rozmieszczenie rdzenia monitorującego oraz interfejsu użytkownika na różnych urządzeniach fizycznych. Umożliwienie rozdzielenia tych dwóch bytów wymaga zapewnienia im wspólnego miejsca, w którym składowane będą dane konfiguracyjne, dane zawierające bieżący stan sieci oraz reprezentację powstałych zdarzeń. System Icinga wykorzystuje do tego celu relacyjną bazę danych. Klasyczny interfejs nie wspiera komunikacji poprzez bazę danych, dlatego należy wykorzystać interfejs icinga-web. Zapewnienie współpracy rdzenia monitorującego z baza danych odbywa się poprzez

Rysunek 4.5. Schemat minimalnej konfiguracji systemu Icinga.

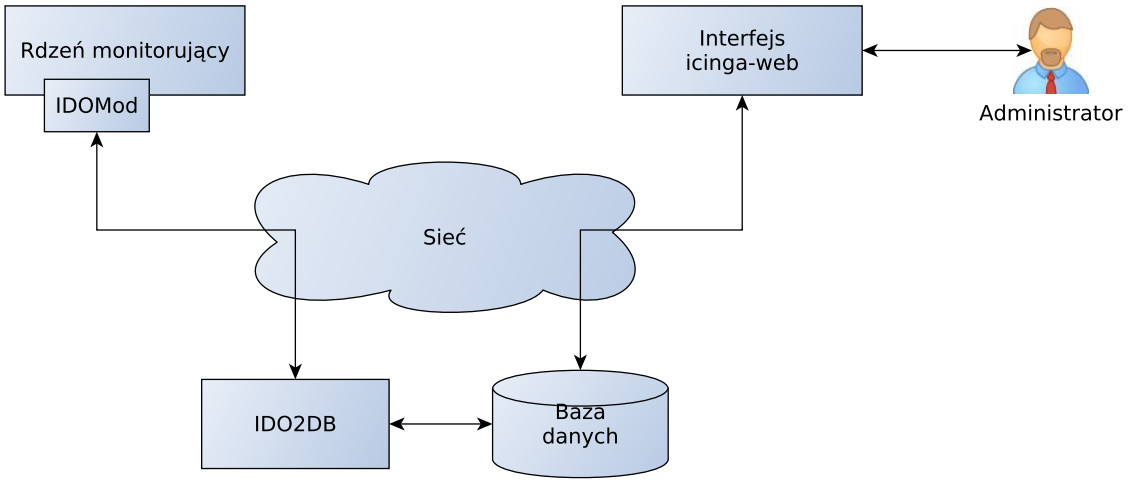


komponent IDOUtils opisany w 4.2. System składa się zatem z następujących elementów:

— rdzeń monitorujący
— baza danych
— interfejs graficzny

Logiczny schemat konfiguracji został przedstawiony na 4.6. Dzięki modularnej budowie całego systemu możliwe jest umieszczenie każdego z wymienionych elementów na osobnym urządzaniu fizycznym. Umożliwia to odciążenie urządzenia, na którym uruchomiony jest rdzeń monitorujący. Ponadto zwiększone zostaje bezpieczeństwo całego rozwiązania, gdyż konieczne jest udostępnienie na zewnątrz jedynie serwera na którym znajduje się interfejs sieciowy. Urządzenie to musi mieć dostęp do bazy danych, lecz nie musi mieć dostępu do urządzenia, na którym umieszczony jest rdzeń monitorujący oraz do całej monitorowanej infrastruktury. Pozwala to na umieszczenie rdzenia monitorującego razem z monitorowaną infrastrukturą za zaporą ogniową, co ogranicza możliwości ingerencji w system monitorowania i infrastrukturę sieciową.

Rysunek 4.6. Schemat podstawowej konfiguracji systemu Icinga.

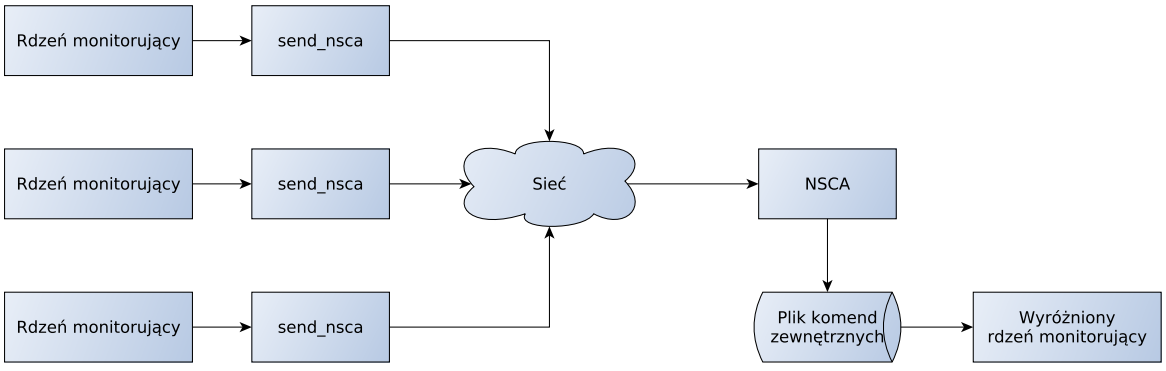


Przedstawiona architektura stanowi bardzo dobrą konfigurację dla firm posiadających jednolitą infrastrukturę sieciową o średniej wielkości. Istnieją jednak sieci

dla których przedstawiona architektura może okazać się niewystarczająca. Jedna z takich sytuacji ma miejsce, gdy instytucja posiada sieć złożoną z kilku segmentów czy to ze względu na separacje czy też lokalizacje geograficzną. Przedstawiona architektura nie umożliwia monitorowania aktywnego, urządzeń znajdujących się za zaporą ogniową. Możliwe jest monitorowanie pasywne takich usług jednak wymaga ono ingerencji w monitorowane serwery. Kolejna z sytuacji ma miejsce, gdy monitorowana infrastruktura, jest na tyle rozbudowana, że urządzenie na którym uruchomiony jest rdzeń nie posiada wystarczającej ilości zasobów, aby monitorować wszystkie urządzenia i usługi. Obie te sytuacje wymagają monitorowania przy jednoczesnym użyciu wielu instancji rdzenia monitorującego.

Pierwszy z możliwych scenariuszy współpracy wielu instancji rdzenia monitorującego wymaga zastosowania dodatku NSCA omówionego w 4.4. Konfiguracja ta zakłada istnienie jednej wyróżnionej instancji rdzenia monitorującego, która będzie odpowiedzialna za przetwarzanie wszystkich wyników sprawdzeń, a także generacje zdarzeń i powiadomień. Konieczne jest również zapewnienie możliwości komunikacji z co najmniej jednego urządzenia w każdym segmencie sieci. Konfiguracja ta została oparta o mechanizm pasywnego sprawdzana usług i urządzeń. Instancja centralna posiada wszystkie usługi skonfigurowane w taki sposób, aby możliwe było dostarczanie pasywnych wyników sprawdzeń tych usług. Na tym samym systemie, co wyróżniona instancja rdzenia uruchomiony jest również serwis systemowy NSCA, który oczekuje na dane przesyłany z instancji roboczych. Każda z instancji roboczych może zarówno wykonywać monitorowanie aktywne jak i pasywne pewnej części usług lub urządzeń. Wyniki sprawdzeń nie są jednak przetwarzane przez instancję roboczą, lecz są przesyłane z użyciem send_nsca do instancji centralnej, w której następuje odpowiednie przetwarzanie. Schemat współpracy poszczególnych elementów systemu w tej konfiguracji zawarto na 4.7.

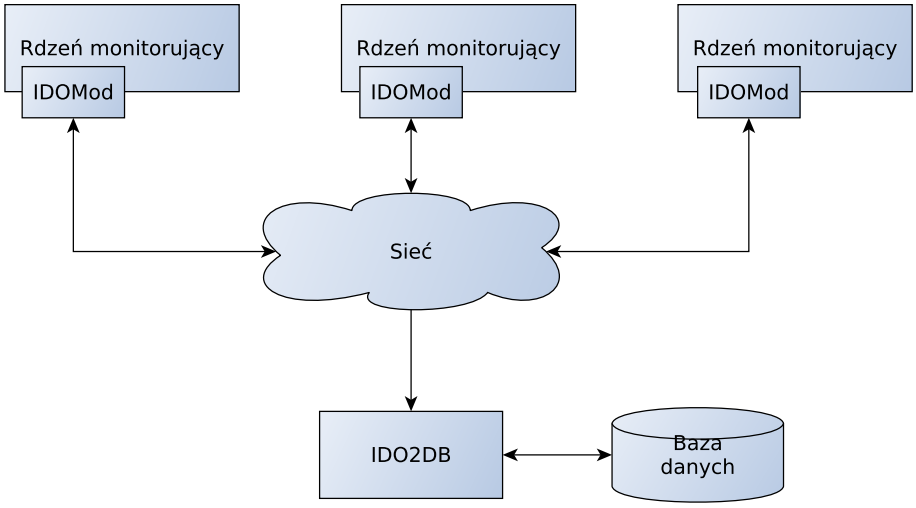
Rysunek 4.7. Schemat konfiguracji rozproszonej z instancją centralną.



Kolejnym z możliwych scenariuszy współpracy wielu instancji rdzenia monitorującego jest wykorzystanie wspólnej bazy danych. Rozwiązanie to wymaga jedynie, aby wszystkie instancje rdzenia miały dostęp do jednej bazy danych. Wszystkie instancje są w pełni niezależne i każda z nich monitoruje w dowolny sposób pewną grupę usług i urządzeń. Wyniki monitorowania są przetwarzane, przez każda instancję niezależnie, a na podstawie ich przetwarzania generowane są odpowiednie zdarzenia. Przy użyciu komponentu IDOUtils wszystkie te dane są konsolidowane

w wspólnej bazie danych z której korzysta interfejs icinga-web. Dzięki wykorzystaniu nowego interfejsu możliwe jest równoczesna prezentacja wyników monitorowania pochodzących od wielu instancji, przy użyciu jednego interfejsu. Logiczny schemat tej konfiguracji przedstawiono na **??**.

Rysunek 4.8. Schemat konfiguracji rozproszonej ze wspólną bazą danych.



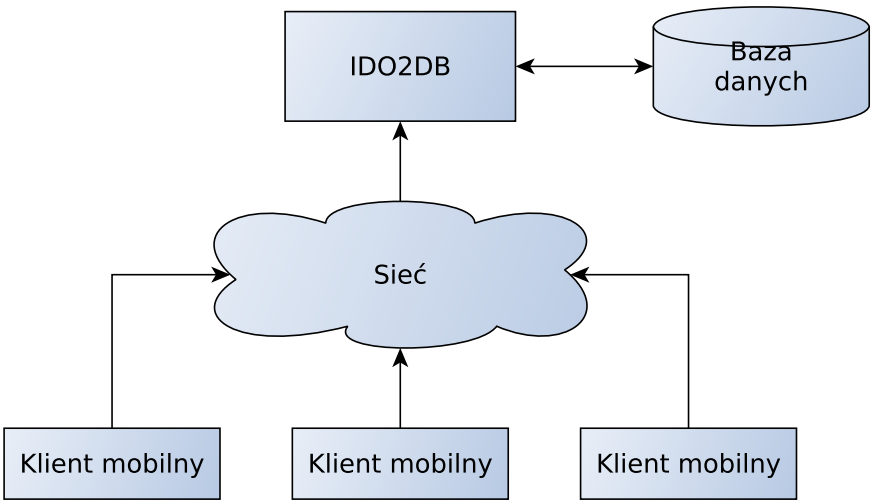
Oba rozwiązania posiadają zarówno zalety jak i wady. Rozwiązanie z użyciem dodatku NSCA zapewnia spójne przetwarzanie danych przez jedna instancje i łatwość konfiguracji dodatków wykorzystujących dane eksportowane przez jądro w postaci danych wydajnościowych. Niestety rozwiązanie to generuje znaczące obciążenie instancji centralnej, gdyż musi ona przetwarzać wszystkie wyniki sprawdzeń. Ponadto należy przypomnieć, że model bezpieczeństwa dodatku NSCA posiada poważne wady. Rozwiązanie oparte o wspólną bazę danych posiada rozproszony mechanizm przetwarzania sprawdzeń jak i zdarzeń dzięki czemu nie występuje w nim nadmierne obciążenie jednej z instancji. Ponadto awaria, dowolnej z instancji nie powoduje nigdy braku możliwości monitorowania całej sieci lecz jedynie jej fragmentu. Niestety w rozwiązaniu tym konieczna jest bardziej zaawansowana konfiguracja dodatków korzystających z danych wydajnościowych. Wybór konfiguracji zależy zatem silnie od infrastruktury w jakiej ma być ona zastosowana, a także od pozostałych elementów systemu, jakie będą wykorzystane.

4.6. Problemy z monitorowaniem klienta mobilnego

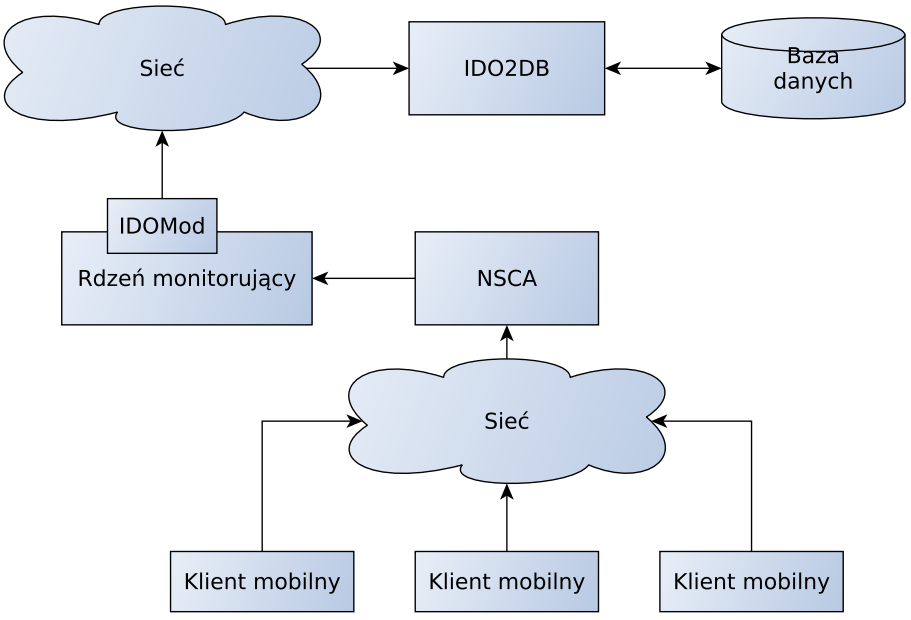
System Icinga nie posiada żadnego mechanizmu wsparcia dla klientów mobilnych. Istnieje wiele konfiguracji rozproszonych, a część z nich może być zaadoptowana do monitorowania klienta mobilnego. Należy pamiętać, iż element systemu obecny na urządzeniu mobilnym musi oszczędzać zarówno pamięć jak i czas procesora. Schemat logiczny konfiguracji ze wspólną bazą danych dla klientów mobilnych został przedstawiony na 4.9. Konfiguracja ta jest niestety nieakceptowalna ze względu na konieczność przetwarzania wszystkich informacji na urządzeniu mobilnym, co w znaczący sposób zwiększyłoby obciążenie klienta mobilnego. W związku z powyższym zdecydowano się rozważyć konfigurację rozproszoną z użyciem NSCA.

Wymaga ona dostarczenia elementu systemu, który będzie znajdował się na urządzeniu mobilnym i monitorował je, a następnie przekazywał, gdy będzie to możliwe dane do instancji nadrzędnej, która będzie prowadziła analizę otrzymanych danych. Schemat tej konfiguracji został przedstawiony na 4.10.

Rysunek 4.9. Monitoring klienta mobilnego w konfiguracji ze wspólną bazą danych.



Rysunek 4.10. Monitoring klienta mobilnego w konfiguracji z instancją nadrzędną.



Wykorzystanie do celu komunikacji pomiędzy klientem mobilnym, a instancja nadrzędną dodatku NSCA niesie za sobą wiele problemów. Dodatek NSCA jest powszechnie używany do monitorowania serwerów znajdujących się za zaporą lub w wydzielonym segmencie sieci. Dodatek ten może być stosowany, w sieciach o statycznym charakterze, gdzie połączenia są stałe, a łączność nie ulega częstym

przerwaniom. Ponadto należy być świadomym słabości modelu bezpieczeństwa stosowanego w protokole wymiany danych. Stosowanie dodatku NSCA poza zamkniętymi sieciami firmowymi może okazać się niebezpieczne i zawodne.

Zagadnienie monitorowania klienta mobilnego zostało szczegółowo opisane w 3. Niestety dodatek NSCA nie spełnia bardzo wielu z przedstawionych wymagań przez co nie powinien być on stosowany w systemach tego typu. Głównymi problemami, które dyskryminują ten w zastosowaniu do monitorowania klienta mobilnego są:

Bezpieczeństwo Mechanizmy bezpieczeństwa zawarte w protokole wymiany danych posiadają poważne luki. Zastosowanie CRC32 do sprawdzania spójności danych niesie za sobą ryzyko ze względu na duże prawdopodobieństwo wystąpienia kolizji. Ponadto konieczność przechowywania na urządzeniu klucza symetrycznego, którego ujawnienie kompromituje cały system znacząco osłabia stosowane mechanizmy bezpieczeństwa.

Nadpisywanie stempla czasu Moduł odbierający dodaje do każdego wpisu dziennika aktualny stempel czasy. Powoduje to brak możliwości przesyłania, historycznych danych zgromadzonych w skutek utraty dostępu do sieci.

Brak dodatkowych mechanizmów uwierzytelnienia klienta Decyzja o przydzieleniu klientowi dostępu czyli akceptacji przesłanych przez niego wpisów dziennika podejmowana jest na podstawie znajomości przez niego algorytmu szyfrowania oraz klucza.

Brak kontroli otrzymywanych danych Każdy klient, który zna klucz może przesyłać wpisy dotyczące dowolnego urządzenia i dowolnej usługi. Brak jest mechanizmu, który pozwolił by na kontrolę tego, jaki klient ma prawo informować o jakim urządzeniu czy też usłudze.

Brak potwierdzenia dostarczenia danych Klient wysyłający dane nie ma żadnej informacji o tym, czy jego dane zostały zaakceptowane czy odrzucone. Oznacza to brak możliwości synchronizacji danych na kliencie mobilnym i serwerze, gdyż nigdy nie mamy gwarancji, że wysłane przez klienta dane zostały przetworzone przez dodatek NSCA i przekazane do rdzenia monitorującego.

Brak implementacji dla systemów mobilnych Moduł wysyłający jest aktualnie zaimplementowany jedynie na systemy Windows oraz Linux. Wiele współczesnych urządzeń mobilnych, które powinny być monitorowane funkcjonuje pod kontrolą systemu operacyjnego Android czy też Windows Phone.

Przekazywanie danych tylko w jedno miejsce Dane odebrane przez moduł odbierający mogą być przekazane jedynie w jedno miejsce. Przy bardziej złożonych systemach, konieczna jest możliwość przekazywania danych do kilku systemów oraz definiowania reguł, które dane gdzie powinny trafić.

Zastosowanie konfiguracji z nadrzędną instancją rdzenia monitorującego stanowi dobry szkielet dla systemu monitorowania klienta mobilnego. Niestety dostępne na rynku narzędzie, to jest dodatek NSCA nie są odpowiednio przystosowane do użycia ich w takim systemie. Wobec braku dostępnych narzędzi na rynku konieczne jest zaprojektowanie oraz zaimplementowanie nowego narzędzie, które spełni stawiane przed nim wymagania.

5. Projekt systemu

5.1. Podział na moduły

Brak dostępnego na rynku systemu wspierającego monitorowanie klienta mobilnego powoduje konieczność opracowania nowego rozwiązania, które spełni wszystkie wymagania opisane w 3. Opracowanie od podstaw nowego systemu monitorowania, wymaga bardzo dużych nakładów pracy. Na rynku obecne są systemy monitorowania klienta statycznego spełniające znaczną część wymagań. Implementacja zatem nowego systemu monitorowania jest nieuzasadniona ekonomicznie oraz merytorycznie. Na podstawie wyników analizy dostępnych na rynku systemów podjęto decyzję, aby wykorzystać system monitorowania Icinga.

Zastosowanie systemu Icinga pozwala na uzyskanie niskim nakładem pracy, wielu funkcjonalności niezbędnych w projektowanym systemie. System monitorowania Icinga jest jednym z najpopularniejszych narzędzi służących do monitorowania infrastruktury statycznej. Posiada on bardzo wiele konfiguracji rozproszonych, zatem możliwe jest monitorowanie nawet bardzo rozbudowanej sieci. Wiele dostępnych dodatków pozwoli również na zapewnienie możliwości analizy danych zarówno historycznych jak i bieżących. Łatwo zatem zauważyć, że dzięki zastosowaniu systemu dostępnego na rynku uzyskano realizację znacznej części wymagań. System nie posada jednak żadnych zintegrowanych mechanizmów monitorowania klienta mobilnego. Konieczne jest zatem opracowanie dodatkowych elementów, które pozwolą na monitorowanie klienta mobilnego zgodnie z przedstawionymi wymaganiami.

Klient mobilny, zdefiniowany w 3 jest urządzeniem, co do którego, nie można zakładać, że posiada nieprzerwany dostęp do sieci internet. Ponadto należy zauważyć zmienność zarówno geograficznego miejsca użytkowania jak i topologi wykorzystywanej infrastruktury sieciowej. Dodatkowo, należy odnieść się do wymagań, w których zawarta jest konieczność minimalizowania zużycia energii przez klienta mobilnego. Ciągłe utrzymywanie połączenia z serwerem, powodowałoby znaczne zużycie energii. Współpraca klienta mobilnego z infrastrukturą publiczną nie pozwala również, na założenie, iż klient mobilny posiada globalny adres IP[1]. Wszystko to razem powoduje to brak możliwości wykorzystania mechanizmów aktywnego monitorowania zawartych w systemie Icinga do monitorowania klienta mobilnego.

Brak możliwości inicjowania przez system monitorujący komunikacji wymusza użycie jednej z dwóch dostępnych konfiguracji rozproszonych. Pierwsza konfiguracja, zakłada przekazywanie jedynie rezultatów pomiarów do jednej z instancji rdzenia monitorującego, który następnie dokona ich przetwarzania. Po wykonaniu wszystkich niezbędnych czynności rezultaty zostaną dostarczone do centralnej

[1] Globalny adres IP - adres protokołu internetowego działającego w warstwie sieciowej, pozwalający na unikalną identyfikację urządzenia w ramach całej sieci Internet.

bazy danych. Druga z konfiguracji, zakłada wykonanie całej niezbędnej analizy danych na urządzeniu mobilnym, a następnie przekazanie rezultatów do bazy danych. Charakterystyka klienta mobilnego przedstawiona w 3, określa iż urządzenie mobilne posiada ograniczone zasoby i ilość dodatkowych operacji wykonywanych na nim powinna zostać ograniczona do minimum. Powyższe wymaganie dyskwalifikuje rozwiązanie, które wymaga przetwarzania wyników pomiarów na urządzeniu mobilnym. Konieczne jest zatem wykorzystanie konfiguracji, w której na urządzeniu mobilnym znajduje się narzędzie przeznaczone jedynie do zbierania danych oraz przekazywania ich do nadrzędnej instancji jądra monitorującego. W klasycznym wariancie tej konfiguracji, która wykorzystywana jest podczas monitorowania infrastruktury statycznej, do przekazywania danych wykorzystuje się dodatek NSCA. Przeprowadzona w 4.4 analiza narzędzia NSCA oraz protokołu komunikacyjnego wykazała liczne uchybienia tego narzędzia oraz wykorzystywanego w nim protokołu. Konieczne jest zatem opracowanie metody komunikacji spełniającej przedstawione wymagania wcześniej wymagania. Ponadto wiele ograniczeń narzędzia NSCA spowodowało konieczność zaprojektowania i implementacji nowego narzędzia, które jest wolne od ograniczeń poprzednika. Powyższe czynniki determinują w znacznym stopniu architekturę system. W celu zapewnienia elastyczności projektowanego systemu zastosowano budowę modularną. System monitorowania składa się z następujących modułów:

Moduł podstawowy Zawiera wszystkie instancje rdzenia monitorującego, zarówno te wykorzystywane do monitorowania infrastruktury statycznej, jak i te których zadaniem jest przetwarzanie danych pochodzących od klientów mobilnych. Ponadto w module tym zawiera się interfejs użytkownika wraz ze wszystkimi dodatkami oraz magazyn danych.

Moduł odbioru danych Składa się on z programu, który zapewnia odbiór danych od klienta mobilnego przy zachowaniu wszystkich wymagań zarówno w kwestii bezpieczeństwa jak i funkcjonalności. Ponadto moduł ten odpowiedzialny jest za przekazywanie odebranych danych do pozostałych elementów zgodnie ze zdefiniowaną w systemie polityka.

Moduł mobilny Zależna od platformy aplikacja mobilna, której podstawowym zadaniem jest gromadzenie danych o zadanych parametrach. Zawiera się tu również implementacja protokołu komunikacyjnego dla danej platformy w celu przekazania zebranych danych do pozostałych modułów systemu.

5.2. Projekt modułu podstawowego

Moduł ten składa się z kilku współpracujących ze sobą elementów. Możliwa jest konfiguracja tego modułu w kilku wariantach, co umożliwia dostosowanie go do rozmiarów oraz topologi monitorowanej infrastruktury. Każda z stosowanych konfiguracji musi zapewniać co najmniej poniższą funkcjonalność:

— monitorowanie infrastruktury statycznej,
— przetwarzanie danych pochodzących od klienta mobilnego,
— gromadzenie danych,
— zapewnienie interfejsu dla administratora.

Minimalna konfiguracja modułu musi się składać co najmniej z jednej instancji rdzenia monitorującego oraz dowolnego z interfejsów, klasycznego lub icinga-web.

W celu umożliwienia przetwarzania danych pochodzących od klientów mobilnych konieczne jest jedynie zdefiniowanie tych urządzeń oraz ich usług, jako monitorowane pasywnie. Należy jednak zwrócić uwagę na liczne ograniczenia tej konfiguracji, które zostały omówione w 4. Ponadto konfiguracja ta nie spełnia wszystkich wymagań, gdyż nie umożliwia analizy zgromadzonych danych historycznych. W celu spełnienia wszystkich wymagań należy zatem wzbogacić omawianą konfigurację o dodatek inGraph, który umożliwi administratorowi analizę zgromadzonych danych.

Liczne wady przedstawionej konfiguracji powodują, że jej funkcjonalność jest znacząco ograniczona. Wykorzystanie jej możliwe jest jedynie w bardzo małych sieciach, które nie będą już rozwijane. Zalecana jest zatem konfiguracja o nieco rozszerzonej strukturze. W skład tej konfiguracji wchodzą:

— rdzeń lub rdzenie monitorujące z komponentem IDOUtils
— baza danych systemu Icinga
— dodatek inGraph
— baza danych dodatku inGraph
— interfejs icinga-web

Minimum wymaganym do funkcjonowania tej konfiguracji jest jedna instancja rdzenia monitorującego. Będzie ona monitorowała zarówno infrastrukturę statyczną jak i przetwarzała dane od klienta mobilnego. Wszystkie dane przetworzone przez tą instancję trafiają do bazy danych z której korzysta interfejs użytkownika icinga-web. Taka konfiguracja umożliwia bardzo dobrą skalowalność całego systemu oraz łatwego dostosowania go do struktury monitorowanej sieci. Rozbudowa infrastruktury monitorującej może zostać wykonana w łatwy sposób poprzez dodanie kolejnej instancji jądra wykorzystującej tą samą bazę danych. Ponadto możliwe jest również łączenie tej konfiguracji z niemalże dowolną inną konfiguracją bez zaburzania pracy systemu. Skalowalność tej konfiguracji dotyczy również klienta mobilnego. Jeśli klientów mobilnych jest zbyt dużo, aby mogły zostać obsłużone przez jedną instancję, możliwe jest dodanie kolejnej instancji, która będzie odpowiedzialna za przetwarzanie danych od wyznaczonej części klientów mobilnych. W celu umożliwienia analizy danych historycznych zalecane jest użycie dodatku inGraph. Został on opisany szczegółowo w 4.3. Jego wykorzystanie umożliwia prezentację administratorowi zarówno uśrednionych wartości z długiego okresu czasu jak i szczegółowych danych z zadanego przedziału. W celu wykorzystania tego dodatku w omawianej konfiguracji konieczne jest umieszczenie elementu zbierającego dane przy każdej instancji rdzenia monitorującego. Wszystkie zebrane dane zapisywane są w jednej bazie danych z której korzysta interfejs użytkownika.

5.3. Protokół komunikacyjny

5.3.1. Architektura

Wymagania przedstawione w 3 definiują bardzo wiele cech systemu, które muszą być zapewnione poprzez użycie odpowiedniego protokołu komunikacyjnego. Analiza wymagań pozwoliła na wyodrębnienie następujących cech protokołu komunikacyjnego:

spójność danych protokół musi gwarantować, że dane zostały dostarczone i przetworzone;

integralność protokół musi zapewniać, że dane zostaną dostarczone w niezmodyfikowanej postaci i jedynie od uwierzytelnionego nadawcy;

poufność protokół musi zapewniać przekazanie danych w sposób, który uniemożliwi stroną trzecim ich odczytanie;

niezależność algorytmu szyfrowania protokół musi być niezależny od algorytmu, którym szyfrowane są dane;

uwierzytelnienie klienta protokół musi zapewniać element pozwalający na potwierdzenie tożsamości klienta;

niezależność uwierzytelnienia klienta protokół musi być niezależny od wykorzystywanej metody uwierzytelnienia klienta;

uwierzytelnienie serwera protokół musi zapewniać potwierdzenie tożsamość serwera;

odporność na utratę urządzenia klienckiego protokół nie może wymagać przechowywania na urządzeniu danych pozwalających na kompromitację całego systemu;

oszczędność pasma protokół powinien minimalizować ilość przesyłanych danych.

Rozbudowane wymagania bezpieczeństwa protokołu wynikają z charakterystyki przesyłanych danych. Dane, które pochodzą z urządzenia mogą zawierać zarówno poufne dane właściciela jak i tajemnice handlowe firmy. Ujawnienie tych danych może pociągać za sobą poważne konsekwencje finansowe lub prawne, dlatego konieczne jest zapewnienie bezpiecznego protokołu. Należy również pamiętać, że jedna ze stron komunikujących się przy użyciu protokołu znajduje się na urządzeniu mobilnym przez co należy ograniczyć narzut wprowadzany przez użycie tego protokołu.

Spośród bezpiecznych protokołów komunikacyjnych rozpowszechnionych na rynku najbliższym spełnienia wszystkich wymagań jest protokół Secure Socket Layer - SSL. Jest to protokół warstwy prezentacji, który pozwala na bezpieczny transport strumienia danych. Protokół wykorzystuje zarówno kryptografię symetryczną jak i asymetryczną. Kryptografia asymetryczna wykorzystywana jest do uwierzytelnienia serwera i opcjonalnie klienta przy pomocy certyfikatów nadawanych przez centra certyfikacji. Model bezpieczeństwa zastosowany w tym protokole pozwala przy pomocy kluczy centrów autoryzacji dokonywać weryfikacji certyfikatów przesyłanych przez wiele witryn. Niestety głębsza analiza protokołu wykazała, iż nie spełnia on wszystkich wymagań. Przede wszystkim zestawienie połączenia wymaga przesłania znaczącej ilości danych. Ponadto konieczne jest zdobycie certyfikatu, który pozwalałby na weryfikację serwera. Model bezpieczeństwa zastosowany w SSL jest dla rozpatrywanego przypadku nadmiarowy, ponieważ klient mobilny przekazuje dane zawsze do tego samego serwera. Protokół SSL przeznaczony jest głównie dla sklepów internetowych oraz banków, gdyż jest on nastawiony na uwierzytelnienie serwera i zapewnia bezpieczny kontakt w wieloma domenami przy użyciu niewielkiej liczby centrów certyfikujących.

Nadmiarowość modelu bezpieczeństwa protokołu SSL powoduje nadmierne zużycie pasma. Ponadto zamknięty zbiór algorytmów możliwych szyfrowania możliwych do wykorzystania powoduje, że nie może on być zastosowany w omawianym przypadku. Wobec braku gotowego protokołu konieczne jest zaprojektowanie nowego, który spełni wszystkie stawiane wymagania.

Protokół został oparty na protokole TCP, który zapewnia abstrakcję przesłania strumienia bajtów z gwarancją ich dostarczenia. Mnogość wymagań dotyczących projektowanego protokołu utrudnia wykorzystanie architektury jednowarstwowej. Konieczne jest zatem wydzielenie warstw z których każda będzie zapewniała dobrze zdefiniowane usługi dla warstw wyższych.

5.3.2. Warstwa formowania wiadomości

Najniższa warstwa protokołu komunikacyjnego zbudowana jest bezpośrednio na protokole TCP. Komunikujące się strony w swej architekturze wykorzystują paradygmat programowania zdarzeniowego. Abstrakcja strumienia bajtów zapewniana przez protokół TCP nie jest odpowiednia dla tego modelu. Konieczne jest zatem dostarczenie warstwy, która umożliwi przesłanie w całości komunikatu o zadanej długości. Umożliwia to wygodne przesyłanie wiadomości odpowiadających poszczególnym zdarzeniom w komunikujących się programach.

Usługa zapewniana przez tą warstwę jest bardzo prosta, dzięki czemu rozpoczęcie komunikacji nie wymaga żadnej inicjalizacji. Protokół jest w pełni symetryczny. Oznacza to, że obie komunikujące się strony posiadają taki sam dozwolony zbiór stanów protokołu. Warstwa świadczy usługę przekazywania wiadomości o zdefiniowanym rozmiarze. W celu wykonania tej usługi, do danych, które są dostarczone stronie nadawczej dołączana jest ich długość. Długość w protokole jest reprezentowana jako 32 bitowa liczba ze znakiem o sieciowej kolejności bajtów. Tak sformatowana wiadomość przesyłana jest przy użyciu protokołu TCP do odbiorcy. Odbiorca po odebraniu pierwszych czterech bajtów wiadomości sprawdza rozmiar danych, po czym rozpoczyna odbieranie ilości danych określonej przez nagłówek. Wiadomość jest przekazywana użytkownikowi dopiero w momencie odebrania całego komunikatu. Jeśli w trakcie odbierania fragmentów wiadomości nastąpi przerwanie połączenia, odebrane fragmentu wiadomości są porzucane, a użytkownikowi sygnalizowany jest błąd.

Długość danych	Dane

Tablica 5.1: Struktura komunikatu warstwy formowania wiadomości

5.3.3. Warstwa kryptograficzna

Warstwa ta jest odpowiedzialna za zapewnienie poufności oraz integralności przesyłanych danych. Ponadto zadaniem tej warstwy jest również wykonanie uwierzytelnienia serwera. Podczas projektowania tej warstwy konieczne było uwzględnienie również wymagania, które zalecało niezależność algorytmu szyfrowania przesyłanych danych od protokołu komunikacyjnego. W celu zapewnienia możliwości późniejszej modyfikacji protokołu, warstwa ta zawiera również proces negocjacji wersji protokołu.

Model bezpieczeństwa implementowany przez tą warstwę jest zbliżony do protokołu SSL, jednak zostały wprowadzone zmiany, które zmniejszają zużycie pasma oraz eliminują potrzebę wykorzystania certyfikatów. Zanim możliwa będzie bezpieczna komunikacja z użyciem tej warstwy konieczne jest umieszczenie na urządzeniu mobilnym klucza publicznego RSA oraz klucza prywatnego na serwerze. Istotne jest, aby klucze mogły być umieszczane jedynie przez autoryzowaną osobę,

np. administratora tych urządzeń. Bezpośrednie umieszczenie klucza publicznego serwera na urządzeniu eliminuje potrzebę wykorzystania certyfikatów oraz ich przesyłania. Należy jednak zwrócić uwagę, iż w przyjętym modelu nie jest możliwa zmiana klucza publicznego serwera bez ponownego umieszczenia go na urządzeniu. Nie stanowi to jednak problemu, gdyż zmiana klucza publicznego w projektowanym systemie jest sytuacją niezwykle rzadką. Ponieważ szyfrowanie asymetryczne wymaga znacznie większego narzutu obliczeniowego jest ono używane tylko w trakcie nawiązywania połączenia. Właściwy transport danych szyfrowany jest przy pomocy uzgodnionego klucza symetrycznego.

Komunikacja z użyciem tej warstwy możliwa jest dopiero po wykonaniu inicjalizacji. Proces inicjalizacji rozpoczyna się od negocjacji używanej wersji protokołu. Niezwłocznie po nadejściu połączenia od klienta serwer wysyła komunikat będący zapytaniem o żądaną przez klienta wersję protokołu. Klient odpowiada na ten komunikat przesyłając żądaną wersję protokołu. Jeśli serwer może obsłużyć dana wersję protokołu przesyła on pozytywne potwierdzenie do klienta, co powoduje rozpoczęcie kolejnego etapu inicjalizacji. W przeciwnym przypadku serwer przesyła informację o odrzuceniu żądania. Po odebraniu negatywnego potwierdzenia klient może podjąć kolejne próby używając innych wersji protokołu. Dalsza komunikacja uzależniona jest od wybranej wersji protokołu komunikacyjnego.

Kod REQUEST_PROTOCOL	Nazwa wersji

Tablica 5.2: Struktura komunikatu żądanie wersji

W zaproponowanej w tej pracy wersji protokołu kolejnym etapem inicjalizacji połączenia jest uwierzytelnienie serwera połączone z przedstawieniem klienta. Etap ten rozpoczyna się od przesłania przez klienta jego identyfikatora oraz losowych ośmiu bajtów danych. Komunikat zaszyfrowany jest kluczem publicznym serwera, zatem może go odczytać jedynie posiadacz odpowiedniego klucza prywatnego. Serwer po odebraniu wiadomości odszyfrowuje ją używając swojego klucza prywatnego. Jeśli wiadomość jest nieczytelna lub klient o podanym identyfikatorze nie istnieje serwer niezwłocznie kończy połączenie. Jeśli wiadomość jest poprawna, a klient o podanym identyfikatorze istnieje serwer wykonuje podpis cyfrowy identyfikatora oraz losowych bajtów odebranych od klienta. Do klienta odsyłany jest komunikat zawierający wykonany przez serwer podpis. Klient po odebraniu podpisu wykonuje weryfikację podpisu na podstawie wiadomości, która została przesłana do serwera. Jeśli podpis jest zgodny oznacza to, że urządzenie z którym nastąpiło połączenie jest posiadaczem odpowiedniego klucza prywatnego, czyli upoważnione przez administratora do odbierania danych pochodzących od tego klienta.

Kod CLIENT_ID	Ciąg losowy	Identyfikator

Tablica 5.3: Struktura komunikatu identyfikatora klienta

Kod CHOOSE_ALGORITHM	Podpis danych z CLIENT_ID

Tablica 5.4: Struktura komunikatu potwierdzającego identyfikator

Ostatnim etapem inicjalizacji komunikacji w tej warstwie jest negocjacja algorytmu szyfrowania oraz generacja odpowiedniego klucza symetrycznego. Klient

przesyła do serwera zaszyfrowany kluczem publicznym komunikat zawierający żądanie algorytmu symetrycznego. Serwer po odebraniu komunikatu odszyfrowuje go przy użyciu klucza prywatnego. W zależności od dostępności żądanego algorytmu, do klienta odsyłany jest komunikat akceptujący lub odrzucający wybrany algorytm. Klient po odebraniu negatywnego potwierdzenia może ponownie zażądać innego algorytmu symetrycznego. Klient po odebraniu komunikatu akceptującego wybrany algorytm dokonuje generacji klucza symetrycznego, a następnie oblicza jego skrót. Tak przygotowany komunikat szyfrowany jest kluczem publicznym i przesyłany do serwera. Serwer po odebraniu klucza sprawdza jego poprawność oraz zgodność z dołączonym skrótem, co stanowi ostatni etap inicjalizacji.

Kod CHOSEN_ALGORITHM	Nazwa algorytmu

Tablica 5.5: Struktura komunikatu żądania algorytmu

Długość skrótu	Kod ESTABLISH_ENCRYPTION	Klucz symetryczny	Skrót

Tablica 5.6: Struktura komunikatu zawierającego klucz symetryczny

Wykonanie inicjalizacji pozwoliło na uzgodnienie w sposób bezpieczny klucza symetrycznego. W dalszej komunikacji wszystkie komunikaty szyfrowane są z użyciem wybranego algorytmu symetrycznego. Ponieważ algorytmy szyfrowania nie zapewniają integralności przesyłanych danych konieczne jest użycie funkcji skrótu. W omawianym protokole została użyta funkcja SHA2. Przygotowanie zatem każdego komunikatu z danymi rozpoczyna się od obliczenia skrótu danych. Ponieważ algorytm skrótu nie był negocjowany konieczne jest dostarczenie długości używanego skrótu. Długość skrótu wyrażona w bajtach dołączana jest na początku wiadomości, natomiast sam skrót na końcu. Komunikat przed wysłaniem szyfrowany jest uzgodnionym kluczem symetrycznym.

Długość skrótu	Dane	Skrót danych

Tablica 5.7: Struktura komunikatu danych warstwy kryptograficznej

5.3.4. Warstwa transportu pomiarów

Warstwa ta zapewnia zapewnia transport wpisów dziennika w pakietach o dowolnym rozmiarze. Wykorzystanie warstw niższych gwarantuje zarówno poufność jak i integralność przesyłanych komunikatów. Żadna z niższych warstw nie zapewnia jednak uwierzytelnienie klienta, dlatego jest to również jedno z zadań tej warstwy.

Inicjalizacja komunikacji w tej warstwie rozpoczyna się od negocjacji algorytmu uwierzytelnienia klienta. Serwer przesyła do klienta komunikat informujący o konieczności wyboru algorytmu uwierzytelnienia. Klient przesyła komunikat zawierający nazwę algorytmu, który ma być użyty do potwierdzenia tożsamości. Serwer po otrzymaniu komunikatu sprawdza czy żądany algorytm jest dostępny dla tego klienta. Jeśli nie jest, przesyłany jest komunikat informujący o odrzuceniu żądania, a klient może ponowić żądanie używając innego algorytmu. Jeśli algorytm wybrany przez klienta jest dostępny rozpoczyna się proces uwierzytelnienia.

Kod CHOSEN_AUTH_MODULE	Nazwa algorytmu uwierzytelnienia

Tablica 5.8: Struktura komunikatu żądania algorytmu uwierzytelnienia

Uwierzytelnienie klienta wykonywane jest poprzez zewnętrzne moduły, gdyż protokół komunikacyjny musi być niezależny od protokołu komunikacyjnego. Algorytm uwierzytelnienia uprawniony jest do przesyłania dowolnych danych w obie strony. Jeśli algorytm uwierzytelnienia odrzuci klienta oznacza to natychmiastowe zamknięcie połączenia. Pomyślne zakończenie procesu uwierzytelnienia oznacza, konieczność wykonania sprawdzenia, czy klient posiada zdefiniowane miejsca do których może przekazywać swoje dane. W przypadku braku takiego miejsca, aby dane nie zostały utracone do klienta wysyłany jest komunikat negatywnego potwierdzenia, a połączenie jest zamykane. Jeśli co najmniej jedno miejsce docelowe zostało zdefiniowane do klienta wysyłany jest komunikat informujący o oczekiwaniu na przesłanie danych, co kończy proces inicjalizacji.

Kod AUTH_DATA	Dane

Tablica 5.9: Struktura komunikatu z danymi uwierzytelnienia

Dane przesyłane przez klienta znajdują się w pakietach. Po odebraniu każdego pakietu i jego pomyślnym przetworzeniu przez aplikację obliczany jest skrót danych w nim zawartych. Obliczony skrót jest następnie przesyłany w pakiecie potwierdzającym przetworzenie danych.

Kod LOGS_PORTION	Dane pomiarów

Tablica 5.10: Struktura komunikatu zawierającego dane

Pakiet z danymi może zawierać dowolną liczbę wpisów dziennika. Każdy wpis powinien posiadać odpowiedni format. W szczególności każdy wpis powinien zawierać bajt danych, który pozwala na określenie czy dotyczy on urządzenia czy usługi. Niezbędne jest również przesłanie stempla czasu, który determinuje kiedy dany wpis został utworzony. Stempel ten powinien być zgodny z stemplem czasu systemu Unix oraz być przesłany w sieciowej kolejności bajtów. Do prawidłowego przekazania wpisu do systemu monitorującego konieczne jest również dostarczenie nazwy urządzenia oraz usługi. W celu umożliwienia oddzielenia tych danych konieczne jest zakończenie każdego z tych pól znakiem zerowym. Każdy wpis powinien również zawierać bajt informujący o stanie w jakim jest dana usługa. Kod ten powinien być zapisany przy użyciu jednego bajtu w sposób zgodny z kodami akceptowanymi przez system monitorujący.

Kod HOST	Stempel czasu	Nazwa urządzenia	Kod stanu	Wpis

Tablica 5.11: Struktura pojedynczego wpisu dziennika urządzenia

Kod SERVICE	Stempel czasu	Nazwa urządzenia	Nazwa usługi	Kod stanu	Wpis

Tablica 5.12: Struktura pojedynczego wpisu dziennika usługi

Jeśli odebrane dane posiadają nieprawidłową strukturę, lub klient dokonał próby przesłania danych, do których przesyłania nie ma uprawnień połączenie jest natychmiast zamykane. Klient, po przesłaniu wszystkich danych lub w chwili zdefiniowanej przez administratora może zdecydować o zamknięciu połączenia wysyłając odpowiedni komunikat. Serwer po odebraniu tego komunikatu zamknie połączenie.

5.4. Projekt modułu mobilnego

Moduł ten jest odpowiedzialny za monitorowanie zadanych parametrów urządzenia mobilnego. Klienty mobilne są wzajemnie nie zależne i mogą operować bez możliwości komunikacji pomiędzy sobą. Konieczne jest zatem, aby każdy klient mobilny posiadał swoją instancję tego modułu. W module tym możemy wyróżnić trzy podstawowe, rozdzielne funkcjonalnie elementy:

— element pomiarowy,
— element komunikacyjny,
— zestaw wtyczek.

Element pomiarowy jest odpowiedzialny za planowanie i wykonywanie pomiarów zgodnie z polityką zdefiniowaną przed administratora. Ponadto konieczne jest zapewnienie składowania uzyskanych informacji do czasu udanej synchronizacji z serwerem. Element komunikacyjny jest to implementacja opisanego wcześniej protokołu komunikacyjnego dla danej platformy. Zadaniem tej części jest dostarczenie wyników pomiarów do miejsc zdefiniowanych przez administratora. Wtyczki są to elementy bezpośrednio odpowiedzialne za wykonywanie pomiarów zadanych wartości czy testowanie zdefiniowanych usług. Metoda realizacji wtyczek uzależniona jest od platformy sprzętowej na jaką przeznaczona jest dana implementacja modułu. Konieczne jest jednak zapewnienie niezależności elementu pomiarowego od zestawu wykorzystywanych wtyczek, aby umożliwić swobodną zmianę zbioru wykorzystywanych wtyczek.

Implementacja tego modułu musi uwzględniać uwarunkowania sprzętowe jak i systemowe platformy na której się znajduje. Urządzenia mobilne są zazwyczaj zasilane z własnych akumulatorów dlatego konieczne jest zastosowanie mechanizmów, które pozwolą na zredukowanie zużycia energii związanego z systematycznym wykonywaniem sprawdzeń. Należy również wspomnieć, iż moduł mobilny odpowiedzialny jest za nadawanie każdemu z odczytów stempla czasu uniwersalnego[2] dokonywanego pomiaru. Na podstawie dokonanej charakterystyki klienta mobilnego, można poczynić założenie, iż klient posiada dostęp do punktów synchronizacji czasu. Jest wiele dostępnych metod synchronizacji czasu na urządzeniu mobilnym, między innymi pobranie czasu z sieci GSM czy też z serwerów czasu światowego, przez co nie stanowi to dla klienta mobilnego poważnego wymagania.

Klient mobilny po zebraniu porcji wpisów dziennika o rozmiarze zgodnym z polityką administratora, lub po upływie określonego czasu powinien przesłać posiadane wpisy dziennika do modułu odbiorczego, a po uzyskaniu potwierdzenia usunąć je z urządzenia w celu oszczędności pamięci. Możliwa jest również sytuacja, w której klient mobilny użytkowany jest przez pewien czas bez dostępu do sieci

[2] Czas uniwersalny - średni astronomiczny czas słoneczny na południku zerowym.

przez którą możliwa jest komunikacja z serwerem. W takiej sytuacji moduł mobilny powinien gromadzić odczyty, aż do czasu uzyskania możliwości połączenia z serwerem. Różnorodność platform dostępnych na rynku sprawia, iż nie można wymagać od modułu odbiorczego dostarczenia uniwersalnej implementacji protokołu komunikacyjnego. Wymaga się zatem, aby klient mobilny używał protokołu komunikacyjnego zgodnego z protokołem modułu odbiorczego. Konieczne jest również, aby klient mobilny posiadał możliwość definiowania metod uwierzytelnienia. Należy również zapewnić możliwość weryfikacji tożsamości serwera, z którym nawiązuje się połączenie.

W ramach systemu monitorowania możliwe jest funkcjonowanie wielu instancji modułu mobilnego. Instancje te mogą być uruchomione na bardzo wielu platformach. W chwili pisania tej pracy nie znaleziono na rynku żadnej aplikacji przeznaczonej, na platformę mobilną, która spełniałaby stawiane wymagania. Szczegółowy projekt oraz implementacja tego modułu dla platformy mobilnej wykracza poza zakres niniejszej pracy. Podczas okresu testowania wykonanego systemu wykorzystano moduł mobilny, przeznaczony dla platformy Android. Został on zaprojektowany i zaimplementowany przez Pana Marcina Kubika. Szczegółowy opis tej implementacji klienta mobilnego można znaleźć w [praca_kubika].

5.5. Projekt modułu odbiorczego

Moduł ten pośredniczy w przekazywaniu danych pomiędzy modułem mobilnym a modułem podstawowym. Uruchomiony jest on na serwerze, który posiada dostęp do zarówno do sieci wewnętrznej instytucji, jak i do sieci, w której funkcjonują klienty mobilne, w szczególności do sieci Internet. Możliwe jest również umieszczenie tego modułu na tym samym urządzeniu, co rdzeń monitorujący odpowiedzialny z przetwarzanie danych pochodzących od klientów mobilnych.

Moduł ten składa się z jednego programu, którego zadaniem jest przekazywanie danych od klientów mobilnych zgodnie ze zdefiniowaną polityką. Znaczna część logiki programu oraz polityka dostarczania danych jest konfigurowana przy pomocy pliku konfiguracyjnego, co umożliwia jej zmianę bez konieczności ponownej kompilacji programu. Plik ten zawiera również definicję klientów, którzy są uprawnieni do przekazywania danych. Każdy klient może posiadać wiele urządzeń, a każde z urządzeń wiele usług. Zgodnie z protokołem komunikacyjnym przed przesłaniem danych występuje etap uwierzytelnienia klienta. Przekazywanie danych możliwe jest jedynie po zakończeniu pozytywnym potwierdzeniu tożsamości klienta. Wysokopoziomowy model logiczny omawianego programu składa się zatem z następujących elementów:

— dostawca danych,
— kanał komunikacyjny,
— konsument danych.

Dostawca danych jest to fragment programu odpowiedzialny za odebranie oraz kontrolę danych pochodzących od klienta. Zawiera on więc implementację protokołu komunikacyjnego oraz wykorzystuje dostępne w programie algorytmy kryptografii oraz uwierzytelnienia zgodnie z konfiguracją. W celu umożliwienia współpracy różnych protokołów komunikacyjnych możliwe jest używanie jednocześnie wielu dostawców danych. Kanał komunikacyjny stanowi natomiast niezawodne

asynchroniczne medium komunikacyjne pomiędzy dostawcą, a konsumentami. Dostawca danych zobowiązany jest do dostarczenia danych jedynie do jednego miejsca, czyli kanału komunikacyjnego. Logika zawarta w kanale wyznacza natomiast na podstawie zdefiniowanej polityki podzbiór konsumentów danych, którzy powinni powinien otrzymać przekazane danych. Możliwe jest definiowanie polityki dostarczania danych zarówno na podstawie informacji o kliencie od którego pochodzą dane, jak i na podstawie informacji o dostawcy który odebrał dane. W celu przyśpieszenia komunikacji z klientem mobilnym potwierdzenie przetworzenia danych zawarte w protokole komunikacyjnym wysyłane jest niezwłocznie po przekazaniu danych do kanału komunikacyjnego. Konieczne jest zatem zapewnienie niezawodności kanału komunikacyjnego, aby możliwe było gwarantowanie, że dane, które do niego zostały przekazane będą dostarczone do wskazanych odbiorców. Odbiorca danych jest to element programu odpowiedzialny za odpowiednie formowanie danych oraz przekazanie ich do modułu podstawowego.

Wymagania przedstawione w 3 wymuszają umożliwienie dodawania nowych algorytmów zarówno kryptograficznych jak i algorytmów uwierzytelnienie klienta. W celu spełnienia tych wymagań wyróżnione zostały w!programie również dwa moduły pomocnicze:

— moduł uwierzytelnienia,
— moduł kryptograficzny.

Moduł uwierzytelnienia stanowi bibliotekę algorytmów uwierzytelnienia klienta, które mogą być wykorzystane przez dostawców danych w celu weryfikacji tożsamości klienta. Moduł kryptograficzny stanowi natomiast bibliotekę algorytmów kryptografii symetrycznej, asymetrycznej oraz funkcji skrótu. Umożliwia to realizację procesu negocjacji algorytmu symetrycznego wykorzystywanego przez protokół komunikacyjny. Ponieważ różnice pomiędzy poszczególnymi protokołami komunikacyjnymi mogą być niewielkie w programie wyodrębniono również moduł implementujący poszczególne warstwy protokołu. Umożliwi to w przyszłości szybką modyfikację protokołu komunikacyjnego lub dodanie nowego.

6. Implementacja

6.1. Moduł odbiorczy

6.1.1. Opis architektury

Logiczna architektura programu została przedstawiona w 5.5. Fizyczna struktura programu została utworzona z użyciem biblioteki Qt jako podstawowego szkieletu aplikacji. Wykorzystano również biblioteki boost oraz Crypto++. Moduł ten przeznaczony jest, podobnie jak system monitorujący Icinga dla komputerów pracujących pod kontrolą systemu operacyjnego Linux i jest uruchamiany jako samodzielny serwis. Fizyczna struktura programu składa się z następujących elementów:

Szkielet programu Zawiera on elementy programu, konieczne do wytworzenia środowiska dla funkcjonowania pozostałych modułów oraz zarządzania nimi. Ponadto zawiera implementacje dostawców oraz konsumentów danych.

Moduł kryptograficzny Dostarcza on implementacji funkcji kryptograficznych, wymaganych podczas komunikacji z klientem mobilnym. Zawiera on zarówno algorytmy asymetryczne, konieczne do inicjalizacji kryptografii symetrycznej, jak i algorytmy symetryczne, służące do przesyłania danych.

Moduł autoryzacji klienta Zawiera on implementację algorytmów uwierzytelnienia klienta.

Moduł komunikacji z użyciem TCP Dostarcza on implementację protokołu komunikacyjnego używanego do komunikacji z klientem.

Moduł logowania Pozwala na przekazywanie użytkownikowi komunikatów z dowolnych miejsc znajdujących się w innych modułach. Wiadomość ta może zawierać informacje o zaistniałym błędzie, lub innym zdarzeniu wymagającym poinformowania użytkownika.

Odwzorowanie podstawowych elementów struktury logicznej w fizyczną ma miejsce w szkielecie aplikacji. Pozostałe elementy programu zostały zaprojektowane jako moduły pomocnicze świadczące dobrze zdefiniowane usługi. Szczególnym przykładem, tego usługowego charakteru pozostałych modułów, może być moduł kryptograficzny i moduł autoryzacji klienta. Udostępniają one generyczne interfejsy dostępu do swoich usług dla pozostałych modułów. Szczegóły implementacyjne są natomiast zamknięte wewnątrz modułów. Typ faktyczny obiektu udostępnianego poprzez generyczny interfejs jest w bardzo wielu przypadkach determinowany dopiero na etapie konfiguracji programu. Ponadto liczba klas znajdujących się w tych modułach może znacząco rosnąć. Jednym z wymagań było zapewnienie możliwości definiowania nowych algorytmów kryptograficznych jak i algorytmów uwierzytelnienia klienta. W celu zapewnienia możliwości wybrania typu faktycznego obiektu

na podstawie danych wykorzystano wzorzec projektowy fabryki[1]. Każdy z omawianych modułów posiada swojego zarządcę. Dostępne w module obiekty muszą zostać zarejestrowane u swojego zarządcy. Pozostałe moduły uzyskują instancje tych obiektów, poprzez zarządcę, który na podstawie przekazanych danych określa typ faktyczny obiektu. Jeśli dany typ jest dostępny, zostanie on wówczas przekazany wywołującemu i będzie on mógł używać go poprzez dostarczony generyczny interfejs. Logiczna struktura tych modułów wymaga zapewnienia, że w programie istnieje tylko jedna instancja obiektu danego zarządcy. W tym celu został wykorzystany wzorzec projektowy nazwie singleton. Zastosowanie wzorca projektowego fabryki pozwala pozostałym modułom, na korzystanie z obiektów, których typ faktyczny jest nieznany w trakcie implementacji oraz kompilacji programu.

W celu konfiguracji programu wykorzystano zewnętrzny plik w formacie XML. Umożliwia to zmianę ustawień programu bez konieczności jego ponownej kompilacji. Plik konfiguracyjny składa się z czterech zasadniczych sekcji:

Sekcja dostawców danych zawiera dane dostawców, którzy mają zostać uruchomieni podczas startu programu. Umożliwia przekazanie dodatkowych informacji do obiektu dostawcy np. adresu IP lub portu na którym powinien on oczekiwać na połączenia.

Sekcja odbiorców danych zawiera dane odbiorców danych, którzy mają zostać uruchomieni podczas startu programu. Umożliwia przekazanie dodatkowych informacji do obiektu odbiorcy danych, takich jak ścieżka do pliku do którego należy zapisywać dane.

Sekcja definicji klientów zawiera definicję klientów oraz grup klientów. Każda definicja klienta składa się z następujących sekcji:

- **Sekcja autoryzacji** zawiera dane o dozwolonych modułach autoryzacyjnych dla danego klienta. Umożliwia także dodatkową konfigurację instancji modułów przeznaczonych dla danego klienta.
- **Sekcja filtrowania** zawiera urządzenia oraz usługi, których dane monitorowania mogą być przesyłane przez tego konkretnego klienta.

Sekcja definicji ścieżek danych zawiera definicję ścieżek danych w programie. Pozwala na definiowanie, do którego odbiorcy danych mają trafić dane odebrane od wskazanego klienta.

Podczas uruchamiania programu, plik konfiguracyjny zostaje przeczytany oraz sprawdzony pod kątem poprawności zarówno składniowej jak i semantycznej. Obiekty obiekty obiekty dostawców, odbiorców oraz algorytmów uwierzytelnienia dostarczane są przez zewnętrznych programistów, zatem prawidłowość ich ustawień nie może być sprawdzana na tym samym etapie. Jest to wykonywane dopiero w trakcie inicjalizacji danego obiektu. Należy zatem zawsze po pomyślnej analizie pliku konfiguracyjnego sprawdzić zawartość pliku dziennika wykonania programu.

6.1.2. Szkielet programu

Moduł ten zawiera podstawowe komponenty programu. Zawarto tutaj wszystkie czynności przygotowawcze związane z wczytaniem konfiguracji oraz utworzeniem obiektów wynikających z niej. Ponadto w module tym znajdują się definicję podstawowych bytów logicznych programu. W zależności od pełnionej funkcji można wyróżnić następujące grupy obiektów:

[1] Wzorzec ten został szczegółowo opisany w XXX

Grupa obiektów konfiguracyjnych zawiera wszystkie obiekty używane zarówno do wczytania parametrów uruchomienia programu z linii poleceń, jak również obiekty odpowiedzialne za dostarczenie do programu konfiguracji zawartej w pliku.

Grupa obiektów producentów danych zawiera generyczny interfejs producenta danych oraz fabrykę, umożliwiającą pozyskiwanie obiektów z tej grupy, a także definicję dostępnych producentów danych.

Grupa obiektów konsumentów danych zawiera generyczny interfejs konsumenta danych oraz fabrykę, umożliwiającą pozyskiwanie obiektów z tej grupy, a także definicję dostępnych konsumentów danych.

Grupa obiektów filtrujących zawiera obiekty pozwalające na kontrolę danych otrzymywanych od klienta

Grupa obiektów kanału komunikacyjnego zawiera obiekty powiązane z kanałem komunikacyjnym pomiędzy producentami danych, a ich konsumentami. Zawiera również mechanizmy formatowania danych oraz bufory przeznaczone na dane oczekujące na przekazanie.

Grupa obiektów zarządzających zawiera zarządce programu oraz obiekty pomocnicze. Wykonywane są tutaj wszelkie czynności, które należy wykonać w trakcie uruchamiania programu, a także tworzenie oraz niszczenie obiektów implementujących elementy logiczne programu.

Głównym członkiem grupy obiektów konfiguracyjnych jest parser pliku konfiguracyjnego. Ponieważ plik konfiguracyjny posiada strukturę pliku XML możliwe było wykorzystanie czytelnika strumienia XML z biblioteki Qt. Klasa ta zapewnia generację znaczników oraz sprawdzanie poprawności składniowej czytanego dokumentu. Umożliwiło to implementację prostego parsera rekursywnie zstępującego, który zajmuje się jedynie sprawdzaniem poprawności logicznej znaczników. Ze względu na strukturę plików XML, nie jest możliwa pełna bieżąca kontrola danych w nim zawartych. Konieczne jest zatem wczytanie pliku konfiguracyjnego i odwzorowanie go w strukturach danych, a następnie wykonanie sprawdzenia spójności oraz poprawności tych danych. Obiekt parsera konfiguracji jest również globalnym obiektem udostępniającym parametry konfiguracji. W obiekcie tym znajdują się wszystkie ustawienia oraz definicje wszystkich obiektów logicznych programu.

Grupa obiektów producentów danych składa się z dwóch głównych elementów. Pierwszym z nich jest klasa implementująca wzorzec fabryki. Pozwala ona pozostałym obiektom na uzyskiwanie instancji obiektu dostawcy danych, bez konieczności znania jego typu faktycznego w trakcie pisania kodu czy też kompilacji. Drugim z elementów jest generyczny interfejs dostawcy danych, który musi być implementowany przez każdego dostawcę. Interfejs ten pozwala na wykonywanie wszystkich niezbędnych operacji. Między innymi na przekazanie konkretnej instancji klasy dodatkowych danych inicjujących takich jak adres sieciowy czy numer portu. Należy również nadmienić, że konieczne było również dostarczenie odpowiedniego mechanizmu rejestracji definiowanych obiektów w fabryce. Istotne jest, aby umożliwić programiście rejestrację nowego typu faktycznego obiektu bez konieczności ingerencji w inne pliki źródłowe. W celu ułatwienia tego procesu zostało opracowane makro, które dzięki wykorzystaniu szablonów dokonuje automatycznej rejestracji nowego typu faktycznego obiektu w fabryce. Dzięki jego wykorzystaniu programista, podczas dodawania nowego dostawcy danych musi zapewnić jedynie deklarację oraz definicję nowego typu. Warto zauważyć, że sposób implementacji tego makra

pozwala na umieszczenie definicji nowego typu w pliku nagłówkowym, który jest włączany do wielu jednostek translacji i nie powoduje to błędów kompilacji ani błędów funkcjonowania procesu rejestracji.

Wydruk 6.1. Definicja dostawcy danych

```
DATA_PROVIDER(NazwaTypu, NazwaRejestrowana)
{
  //deklaracja metod
}
```

W ramach tej grupy obiektów dostarczono również referencyjną implementację dostawcy o nazwie typu DefaultTcpProvider. Stanowi on implementację omówionego wcześniej protokółu komunikacyjnego. W celu zapewnienia lepszej wydajności, logika funkcjonowania tego dostawcy została przeniesiona do osobnego wątku programu. Aby zapewnić elastyczności wykorzystania tego obiektu możliwe jest definiowanie z poziomu pliku konfiguracyjnego zarówno adresu IP i portu wykorzystywanego przez ten obiekt, a także wskazanie pliku zawierającego klucz prywatny.

Grupa obiektów konsumentów danych posiada analogiczną budowę jak grupa producentów danych. Zapewnia ona zarówno fabrykę konsumentów danych jak i generyczny interfejs konsumenta. Rejestracja obiektów w fabryce odbywa się również przy użyciu analogicznego makra. W ramach tej grupy obiektów dostarczono implementację dwóch konsumentów danych. Pierwszy z nich o nazwie typu ToScreenPrinter, spełnia jedynie funkcję kontrolną. Wszystkie dane, które do niego trafią są natychmiast wypisywane w dzienniku wykonania programu. Dostawca typu ToIcingaWriter odpowiedzialny jest natomiast za przekazywanie wszystkich danych do pliku komend zewnętrznych systemu Icinga. Możliwa jest zmiana ścieżki pliku komend zewnętrznych systemu Icinga poprzez plik konfiguracyjny.

Grupa obiektów filtracji pozwala na kontrolę danych otrzymywanych od klienta. Każdy klient posiada zdefiniowany w pliku konfiguracyjnym zbiór urządzeń i usług o których informacje może przesyłać. Obiekty z tej grupy pobierają z analizatora składniowego wspomniane zbiory i dokonują kontroli każdego wpisu dziennika otrzymanego od klienta. Jeśli klienta nadesłał dane dotyczące urządzenia lub usługi do których nie ma on uprawnień, nie będzie możliwe przekazanie ich do konsumentów.

Grupa obiektów kanału komunikacyjnego odpowiedzialna jest za niezawodne przekazanie danych od dostawcy danych do konsumentów według reguł zdefiniowanych w pliku konfiguracyjnym. Zapewnienie niezawodności zostało osiągnięte poprzez implementację bufora kołowego wewnątrz pliku. Każdy dostawca danych posiada swój plik bufora. Na początku tego pliku zapisane są położenia miejsca przeznaczonego do czytania oraz miejsca przeznaczonego do pisania. Dane, które logicznie znajdują się pomiędzy miejscem do czytania, a miejscem do pisania są to dane, które zostały odebrane od klienta lecz nie zostały jeszcze dostarczone do konsumentów. Pomyślnie zweryfikowana przez obiekty filtrujące porcja danych przekazywana jest z użyciem kanału komunikacyjnego do konsumentów danych. Operacja ta odbywa się w dwóch etapach. Pierwszy etap dokonuje zapisu danych do pliku bufora. Jeśli aktualnie nie ma żadnych danych oczekujących na przetworzenie przez konsumentów, nowa porcja danych jest niezwłocznie dostarczana do

odpowiednich obiektów. Jeśli istnieją porcje danych, które oczekują na zapisanie dane zostaną zapisane do pliku i dostarczone, po danych, które nadeszły przed nimi. Należy zwrócić uwagę, że dane zapisywane są na dysku tylko raz, niezależnie od liczby dostawców do których powinny one zostać dostarczone.

Głównym przedstawicielem grupy obiektów zarządzających jest klasa główna programu. Program został napisany zgodnie z metodyką obiektową zatem cała logika wykonania programu została również zamknięta w klasie co uprościło funkcję główną programu do minimum. Klasa ta odpowiedzialna jest za przebieg całości programu. Pierwszą operacją wykonywaną przez tą klasę jest odnalezienie pliku konfiguracyjnego i zlecenie jego wczytania przez parser konfiguracji. Na podstawie informacji uzyskanych w wyniku analizy pliku konfiguracyjnego klasa główna programu pobiera z odpowiednich fabryk wszystkie obiekty producentów oraz konsumentów zdefiniowanych w pliku konfiguracyjnym oraz przekazuje im odpowiednie parametry inicjalizacyjne. Klasa ta jest również odpowiedzialna za prawidłową deinicjalizację oraz destrukcję wszystkich obiektów. Ponadto należy zauważyć, że omawiany program funkcjonuje jako serwis systemowy, zatem konieczne jest również wykonanie w tej klasie wszystkich czynności zalecanych przy uruchamianiu takich serwisów.

6.1.3. Moduł kryptograficzny

Moduł ten dostarcza pozostałym elementom programu implementacji algorytmów kryptograficznych. Dostępne są implementacje następujących schematów algorytmów:

— szyfrowanie symetrycznych,
— szyfrowanie asymetrycznych,
— funkcja skrótu,
— podpis cyfrowy.

Każdy ze schematów posiada zdefiniowany generyczny interfejs. Dostarczanie implementacji danego schematu kryptograficznego odbywa się w sposób analogiczny do dostarczania implementacji dostawców danych. Wszystkie implementacje algorytmów zostają zarejestrowane w fabryce kryptograficznej, która umożliwia uzyskiwanie obiektu o typie określanym na podstawie danych na przykład odebranych od klienta..

W ramach tej pracy dostarczono kilku algorytmów, które były konieczne do zaimplementowania protokołu komunikacyjnego. Jako algorytm symetryczny dostarczona została implementacja algorytmu AES pracującego w trybie wiązania bloków zaszyfrowanych. Protokół komunikacyjny wymagał dostarczenia również asymetrycznego algorytmu RSA. W module zdefiniowano ponadto klasę implementującą generyczny interfejs funkcji skrótu, która dostarcza funkcjonalności algorytmu SHA-2 o długości skrótu 256 bitów. Jeden z etapów protokołu komunikacyjnego wymagał również dostarczenia algorytmu podpisu cyfrowego opartego na algorytmie RSA.

Do implementacji wszystkich algorytmów została wykorzystana biblioteka Crypto++. Jest to popularna biblioteka o otwartych źródłach napisana w języku C++, która w obiektowy sposób udostępnia algorytmy kryptograficzne.

6.1.4. Moduł uwierzytelnienia klienta

Moduł posiada architekturę typową dla modułów usługowych. Głównym elementem jest klasa implementująca wzorzec fabryki oraz generyczny interfejs pozwalający na wykorzystywanie obiektów uzyskanych z fabryki.

Interfejs zdefiniowany dla algorytmów uwierzytelnienia został zaprojektowany z wykorzystaniem mechanizmu sygnałów i slotów z biblioteki Qt. Użycie tego mechanizmu pozwala na znacznie bardziej wydajne wykorzystanie zasobów. Przykładem takiej optymalizacji może być czas gdy klient przetwarza żądanie związanie z uwierzytelnieniem wątek serwera nie musi być wtedy bezczynny lecz może przetwarzać żądania pochodzące od innych klientów. Definicja interfejsu pozwala również na przekazanie do niego dodatkowych ustawień pochodzących z pliku konfiguracyjnego. Każdy algorytm uwierzytelnienia powinien po zakończeniu sukcesem lub porażką procesu uwierzytelnienia klienta wykonać emisję sygnału z interfejsu algorytmu wraz z rezultatem procesu autoryzacji.

Zapewnienie niezależności implementacji algorytmów uwierzytelnienia od wykorzystywanej aktualnie metody komunikacji wymagało zdefiniowania generycznego interfejsu komunikacyjnego, który może być wykorzystywany przez implementacje poszczególnych algorytmów. Implementacja tego interfejsu powinna być dostarczona przez moduł, który jest aktualnie wykorzystywany w programie do komunikacji z klientem.

W ramach pracy zostały również zaimplementowane dwa moduły uwierzytelnienia klienta. Pierwszy z nich nosi nazwę AlwaysAllow i jest to tak zwane uwierzytelnienie puste, czyli zakończone sukcesem dla każdego klienta. Drugi moduł - LoginPass jest to prosta metoda uwierzytelnienia oparta na pobraniu od klienta loginu oraz hasła i porównanie go z danymi dostarczonymi w pliku konfiguracyjnym. Każdy klient posiada w pliku konfiguracyjnym listę dostępnych dla niego algorytmów uwierzytelnienia wraz z danymi jakie powinny być przekazane, aby zapewnić pozytywne wykonanie procesu. Należy zwrócić uwagę, że zaimplementowane algorytmy uwierzytelnienia stanową jedynie przykład i nie powinny być wykorzystywane w systemie produkcyjnym, ponieważ wszystkie hasła oraz nazwy użytkowników przechowywane są jawnym tekstem w pliku konfiguracyjnym.

6.1.5. Moduł komunikacji z wykorzystaniem TCP

Moduł ten zawiera implementację protokołu komunikacyjnego opisanego w 5.3. Inicjacja każdej z warstw protokołu została zaimplementowana dedykowanej klasie lub jeśli proces inicjacji złożony był z kilku rozdzielnych logicznie elementów, każdy element został zaimplementowany w osobnej klasie. W celu umożliwienia każdej z warstw protokołu korzystanie z usług warstw niższych w sposób generyczny wykorzystany został wzorzec dekoratora.

Aby wykorzystać wzorzec dekoratora zdefiniowano generyczny interfejs pozwalający na odczytanie oraz zapisanie komunikatu niezależnie od liczby warstw znajdujących się poniżej. Klasą prostą w tym przypadku jest klasa zawierająca implementację warstwy formowania wiadomości. Klasa ta została oparta na implementacji gniazd TCP pochodzącej z biblioteki Qt. Klasami dekorującymi natomiast są klasy zapewniające szyfrowanie oraz dołączanie skrótu wiadomości.

Wykorzystanie wzorca dekoratora pozwoli w przyszłości na łatwą modyfikację protokołu komunikacyjnego np. poprzez dodanie dodatkowej warstwy. Ponadto

wprowadzenie jednolitego interfejsu pozwoliło na zachowanie prostoty i jednolitości implementacji poszczególnych warstw protokołu komunikacyjnego.

Moduł ten został zaimplementowaniu przy użyciu licznych mechanizmów z biblioteki Qt. Przede wszystkim wykorzystany został moduł sieciowy wspomnianej biblioteki. Dzięki jego użyciu uzyskano dostęp do generycznej implementacji serwera TCP, a także gniazd. Szkielet aplikacji Qt pozwolił na wygodną implementację asynchronicznej komunikacji z użyciem gniazd TCP przy pomocy standardowego dla tego szkieletu mechanizmu sygnałów i slotów. Dzięki temu uzyskano przejrzysty i wydajny kod, który pozwala na obsługę wielu klientów w jednym wątku.

6.1.6. Moduł logowania

Omawiany program wykonywany jest bez interakcji z użytkownikiem. Funkcjonuje on jako serwis systemowy. Docelowo będzie on wykonywany na serwerze, poza sesją jakiegokolwiek użytkownika. W trakcie wykonania programu mogą się zdarzyć sytuacje wymagające poinformowania użytkownika o ich wystąpieniu. Znaczna część z tych informacji stanowi jedynie zapis wykonania programu, jednak mogą występować również informacje o sytuacjach krytycznych, o których użytkownik musi zostać powiadomiony. Konieczne było zatem dostarczenie możliwości przekazywania takich informacji z wielu modułów do jednego, wspólnego miejsca, które stanowi dziennik wykonania programu.

Każdy moduł posiada możliwość przekazywania użytkownikowi wiadomości o różnym priorytecie. Dozwolone są następujące priorytety:

FATAL najwyższy priorytet, wiadomość zawiera komunikat o błędzie, który uniemożliwia dalsze wykonanie programu

ERROR komunikat zawiera informacje o błędzie, który uniemożliwia wykonanie pewnej ścieżki programu

WARNING komunikat zawiera ostrzeżenie o nietypowej sytuacji

DEBUG komunikat zawiera treść pomocną podczas wyszukiwania błędów

INFO komunikat zawiera jedynie treści informacyjne

Podczas kompilacji ustalany jest minimalny priorytet wiadomości, które mają być przekazywane użytkownikowi. Wszystkie wiadomości o priorytecie niższym niż ustalony, nie zostaną zapisane. Ponadto dzięki użyciu mechanizmów opartych o szablony wszystkie komunikaty o priorytecie niższym zostaną rozwinięte do wywołania funkcji pustej. Wywołanie takie zostanie z bardzo dużym prawdopodobieństwem zoptymalizowane przez kompilator.

Przekazanie użytkownikowi treści komunikatu w wielu sytuacjach może nieść zbyt mało informacji. W celu umożliwienia przekazania dodatkowych informacji bez konieczności pisania nadmiernej liczby komend przy każdym komunikacie, opracowana została makrodefinicja, która do każdego komunikatu dołączy aktualny stempel czasu, nazwę pliku w którym znajduje się komunikat, a także nazwę funkcji oraz numer linii. Ponadto komunikat nie musi się składać jedynie z tekstu lecz można go formować w taki sam sposób jak pisać do strumienia.

Wydruk 6.2. Przykładowe wypisanie komunikatu

```
LOG_ENTRY(MyLogger::DEBUG, "komunikat"<<123);
```

Wydruk 6.3. Format komunikatu przekazywanego użytkownikowi

```
[stempel czasu][poziom][plik][funkcja][linia]:komunikat123
```

Ponieważ program nie jest przypisany do żadnego z wirtualnych terminali nie ma możliwości przekazywania wiadomości na standardowe wyjście lub wyjście błędów. Konieczne jest zatem utworzenie pliku, do którego zapisywane będą komunikaty. Należy zwrócić uwagę, że program jako serwis systemowy uruchomiony będzie ze znacznie ograniczonymi prawami, aby podnieść poziom bezpieczeństwa serwera. W związku z powyższym jedynym miejscem, co do którego można założyć, że program będzie miał dostęp jest katalog plików tymczasowych. Każde uruchomienie programu powoduje zatem utworzenie w tym katalogu pliku składającego się z nazwy programu oraz stempla czasu zawierającego czas jego uruchomienia.

7. Testowanie i użytkowanie wykonanego systemu

7.1. Testowanie

7.2. Użytkowanie systemu

8. Podsumowanie

Bibliografia

[1] Agavi documentation. http://www.agavi.org/documentation/tutorial.
[2] Birthday problem.
[3] Cacti project homepage. http://www.cacti.net/.
[4] CGI: Common Gateway Interface. http://www.w3.org/CGI/.
[5] Icinga Version 1.9 Documentation. http://docs.icinga.org/1.9/Icinga_v19_en.pdf.
[6] Introducing JSON. http://www.json.org/.
[7] MCrypt project homepage. http://mcrypt.sourceforge.net/.
[8] Nagios Plugin Development Guidelines. https://nagios-plugins.org/doc/guidelines.html
[9] Nagios project homepage. http://www.nagios.org/.
[10] SNMP Technical Articles - overview. http://www.snmplink.org/articles/overview/.
[11] XML-RPC Specification. http://xmlrpc.scripting.com/spec.html.
[12] GNU General Public License version 2, 1991. http://www.gnu.org/licenses/old-licenses/g
[13] RRDtool documentation, 2012. http://oss.oetiker.ch/rrdtool/doc/index.en.html.
[14] N. Neufeld C. Haen, E. Bonaccorsi. Distributed monitoring system based on Icinga, 2011. http://accelconf.web.cern.ch/accelconf/icalepcs2011/papers/wepmu035.pdf.