Projekt TIN - Dokumentacja Wstępna

Michał Dziekoński
Daniel Obrębski
Jakub Skrętowski
Paweł Szymczyk

1. Treść zadania.

a. System zbierania statystyk częstości zapytań do serwerów FTP (np. z użyciem Netfilter).
   i. System obserwuje predefiniowany zbiór maszyn, w którym inicjuje i zatrzymuje pomiar a następnie zbiera dane. Na każdej obserwowanej maszynie instalowany jest agent, z którym komunikuje się serwer zarządzający. Operator współpracuje z serwerem zarządzającym poprzez: 1) skrypty powłoki, 2) przeglądarkę WWW. Ponadto należy zaprojektować moduł do Wireshark umożliwiający wyświetlanie i analizę zdefiniowanych komunikatów. System może działać w sieci IPv6 i IPv4.

2. Nazwę własną projektowanego systemu.

a. FTP Stattr

3. Przyjęte założenia funkcjonalne i niefunkcjonalne.

a. Założenia funkcjonalne (priorytet)

- Aplikacja Agenta
  - Zbieranie danych o ruchu FTP (1)
  - Komunikacja z aplikacją serwerową, bazowana na gniazdach BSD (1)
  - Reakcje na przychodzące żądania z aplikacji serwerowej (1)
  - Zapisywanie dużych ilości zbuforowanych danych do pliku (3)
  - Zbieranie danych o ruchu w określonych porach dnia, w określonych przedziałach czasu (4)
  - Porzucanie danych starszych niż określony okres czasu (4)
- Aplikacja Serwerowa
  - Komunikacja międzyprocesowa z interfejsem terminalowym (1)
  - Dodawanie, modyfikowanie i usuwanie wpisów z listy obserwowanych serwerów z agentami (1)
  - Komunikacja z aplikacjami agentowymi, bazowana na gniazdach BSD (1)
  - Wysyłanie poleceń i odbieranie danych z serwerów agentowych (1)
  - Analiza zebranych danych z serwerów agentowych (3)
  - Komunikacja z interfejsem przeglądarkowym (2)
  - Zarządzanie periodycznymi i automatycznymi procesami zbierania danych z agentów, bez udziału komend od użytkownika (4)
  - Zapis zebranych danych do pliku (4)
  - Modyfikowanie przedziałów czasu zbierania danych dla określonych maszyn agentowych (4)

    - Modyfikowanie ważności danych dla określonych maszyn agentowych (4)
- Komunikacja przez gniazda / Wtyczka do Wiresharka
    - Przygotowanie specyfikacji wiadomości komunikacyjnych (0)
    - Przygotowanie wtyczki do Wiresharka, testującej poprawność komunikatów (0)
    - Opracowanie scenariuszy testowych/testów automatycznych sprawdzających poprawność komunikacji (1)
- Interfejs konsolowy
    - Komunikacja międzyprocesowa z aplikacją serwerową (1)
    - Wysyłanie prostych komend do aplikacji serwerowej (1)
    - Odbieranie i wyświetlanie wyników prostych komend z aplikacji serwerowej (1)
    - Rozszerzenie listy komend o pobieranie danych statystycznych w prostym formacie tekstowym (3)
- Interfejs przeglądarkowy
    - Komunikacja z aplikacją serwerową (2)
    - Wysyłanie prostych komend do aplikacji serwerowej (2)
    - Odbieranie i wyświetlanie wyników prostych komend z aplikacji serwerowej (2)
    - Rozszerzenie listy komend o pobieranie danych statystycznych (3)
    - Wyświetlanie danych statystycznych w prostej i klarownej reprezentacji listowej, tabelarycznej lub wykresowej (3)
    - Eksport danych do pliku (4)
    - Zapis pobranych danych do pamięci lokalnej przeglądarki (4)
- (Priorytety)
    - 0 - natychmiastowy, wymagany do rozpoczęcia właściwych prac implementacyjnych
    - 1 - ważny, pierwszy etap implementacyjny
    - 2 - średni, drugi etap implementacyjny, rozwojowy
    - 3 - mało ważny, trzeci etap, wykończeniowy w stosunku do właściwych funkcjonalności
    - 4 - opcjonalne, dodanie mile widzianych funkcji, jeśli wystarczy na to czasu

b. Założenia niefunkcjonalne

- Łatwość konfiguracji: Program będzie udostępniał interfejsy konfiguracyjne w postaci ustawień w graficznym panelu sterowania oraz API do skryptowania. Użytkownik powinien być zaznajomiony z tematyką programu przed jego użyciem. Opis API oraz sposób obsługi panelu sterowania dostarczone zostaną w instrukcji obsługi. Szacowana trudność: %.
- Wydajność: Program musi w stanie sprawnie reagować na komendy użytkownika oraz być w stanie płynnie przekazywać informacje do serwera zarządzającego oraz sprawnie komunikować się z serwerami FTP, w razie

potrzeby musi sygnalizować błędy oraz o ile to możliwe spróbować wznowić połączenie.
- Bezpieczeństwo: Program powinien chronić użytkownika przed ewentualnymi szkodami wynikającymi z nieprawidłowej konfiguracji. Potrzebna jest więc walidacja danych wprowadzanych do programu.
- Możliwość rozwoju systemu: Program będzie dostępny jako zamknięta i gotowa aplikacja przeznaczona na wiele stanowisk komputerowych. Aplikacje będzie można modyfikować skryptami w zależności od potrzeb.

4. Wybrane środowisko sprzętowo, programowe i narzędziowe (systemy operacyjne, języki programowania).

- Do prowadzenia komunikacji pomiędzy programem a maszynami na których zbierane są informacje o ruchu zostanie zastosowana aplikacja napisana w języku C++ korzystająca gniazd BSD.
- Do utworzenia interfejsu przeglądarkowego zastosowana zostanie aplikacja napisana w JavaScript używająca Backbone.js, Twitter Bootstrapa, jQuery oraz API WebSockets.
- Skryptowanie (przesyłanie komend z konsoli) odbywać się będzie w języku powłoki systemowej, jako wywołanie aplikacji z odpowiednimi parametrami, bądź jako sekwencja wprowadzanych danych po uruchomieniu programu komunikacyjnego z serwerem.
- Skrypty testujące działanie programu napisane zostaną w języku Python
- Rozwiązanie będzie projektowane dla systemów operacyjnych Linux i Mac OS X

5. Architekturę rozwiązania, tj. ilustrację i opis struktury logicznej systemu (koncepcyjnych bloków funkcjonalnych).

Architektura zostanie podzielona na pięć części składowych:

- Aplikacji agenta działającego w trybie demona, nieustannie zbierającego dane do analizy
- Aplikacji serwerowej, działającej również w trybie “aktywnego” demona, zbierającego dane z agentów i wykonującego lokalne analizy
- Wtyczki do programu Wireshark, która posłuży do sprawdzania poprawności komunikacji między agentami a serwerem
- Interfejsu przeglądarkowego udostępniającego pełny zestaw konfiguracji i poleceń dla aplikacji serwerowej
- Interfejsu konsolowego pełniącego bardziej ubogą i “niegraficzną” wersję interfejsu przeglądarkowego

Serwer i agent zostaną oparte o moduły połączone ze sobą w jedną całość za pomocą kontrolerów. Zarówno agent jak i serwer będą dzieliły między sobą wspólny kod do komunikacji poprzez gniazda, będący modułem łączności. Będzie to swego rodzaju biblioteka wykorzystywana w komunikacji.

Agent zostanie wyposażony w moduł pomiarowy, zbierający dane o ruchu FTP na maszynie docelowej. Dane będą przechowywane w pamięci do czasu komunikacji z serwerem, reakcje na żądania serwera zapewni moduł usługowy.
Serwer zostanie wyposażony w moduł analityczny, wykonujący obliczenia na potrzeby utworzenia statystyk pomiarów; moduł komend, nasłuchujący przychodzących poleceń z terminalu i przesyłający wyniki poleceń na terminal; moduł webowy, nasłuchujący przychodzących poleceń z aplikacji webowej, i przesyłający wyniki do użytkownika; moduł zbiorczy, wywołujący łączność z agentami i zbierający otrzymane dane dla modułu analitycznego.
Wtyczka do programu Wireshark zdefiniuje sposób komunikacji między aplikacją główną a aplikacją Wireshark i określi, jak przeprowadzić obróbkę danych do czytelnego formatu.
Interfejs przeglądarkowy, zbudowany jako “Single Page Application”, udostępni prosty lecz rozbudowany dostęp do wszelkich komend i statystyk zebranych przez serwer. Komunikacja między interfejsem a serwerem będzie odbywać się za pomocą protokołu WebSockets wykorzystującego JSON do przekazywania danych (w czasie rzeczywistym, asynchronicznie; nie mylić WebSockets z gniazdami). Wszystkie dane i statystyki będą prezentowane w przejrzystym, przyjemnym dla oka formacie listowym, tabelarycznym lub wykresowym (o ile użytkownik nie zarząda “czystych danych”, w formacie JSON).
Interfejs konsolowy, zbudowany jako osobny proces, wykorzystując IPC będzie komunikował się z działającą w tle aplikacją serwerową, asynchronicznie wywołując zadane polecenia i oczekując na dane. Rezultatem wywołania polecenia w tym interfejsie będzie wyświetlenie danych na konsoli uzytkownika i oczekiwanie na dalsze komendy do przekazania. Ten interfejs będzie miał jedynie możliwość listowego wyświetlenia danych.

6. Sposób testowania.

- Skrypty w Python - zautomatyzowane testy wykonujące zadane scenariusze testowe i sprawdzające poprawność ich wyników.
- Testy jednostkowe w C++ - użyte zostaną do sprawdzenia najważniejszych modułów aplikacji serwerowej i agenckiej - modułu łączności, modułu pomiarowego, modułu usługowego czy kontrolerów.
- Empiryczne testowanie - przygotowane odpowiednie zestawy scenariuszy testowych, które można odtworzyć w dowolnym późniejszym momencie w celu weryfikacji poprawności działania.

7. Sposób demonstracji rezultatów, tj. scenariusze testów akceptacyjnych do zaprezentowania przy odbiorze projektu.

- Poprawność działania wtyczki do programu Wireshark sprawdzimy poprzez włączenie aplikacji Wireshark, uaktywnienie odpowiedniego, zdefiniowanego ruchu sieciowego oraz prostą analizę otrzymanych danych. Powstanie do tego osobny skrypt lub aplikacja.

- Aby, sprawdzić czy system dobrze wykonuje pomiar, możemy wykorzystać aplikację Wireshark, która na podstawie wysłanych/odebranych pakietów, komunikatów potwierdzi poprawność danych ujawnionych w programie i/lub prostego wykresu.
- Aby zademonstrować obsługę błędów w połączeniu, spróbujemy podłączyć się do nieistniejącego serwera podając niepoprawny adres IP lub będąc odłączonym od sieci Internet.
- Zaprezentujemy dane historyczne zebrane przez program i udowodnimy ich poprawność.
- Aby pokazać dostępne opcje w programie, przeprowadzimy krótką miniprezentację. Pomyślne wykonanie będzie świadczyło o poprawnej obsłudze różnych przypadków.
- W celu zaprzentowania możliwości wykrywania anomalii w obserwowanym ruchu przygotujemy specjalny skrypt zlecający wykonanie wielu żądań jednocześnie do jednej z maszyn obserwowanych. Po wykonaniu skryptu serwer powinien zebrać dane z agenta na tej maszynie, poprawnie rozpoznać atak i wyświetlić dane na jego temat.

8. Podział prac w zespole.

- **Daniel Obrębski -** komunikacja na bazie gniazd, przygotowanie wtyczki Netfilters, moduł zbiorczy dla Serwera
- **Michał Dziekoński** - przygotowanie szkieletu kodu dla Agenta i Serwera, interfejs przeglądarkowy
- **Paweł Szymczyk** - moduł pomiarowy i usługowy dla Agenta, interfejs konsolowy
- **Jakub Skrętowski** - moduł analityczny i komend dla Serwera, wtyczka do Wiresharka