Studentská tvůrčí a odborná činnost
STOČ 2017

Vývoj a nasazení SW prostředků

pro analýzu přístupů do sítě VŠB-TU Ostrava

Jakub KALNIK

Vysoká škola báňská – Technická univerzita Ostrava
17. listopadu 15/2172
708 33 Ostrava-Poruba

20. dubna 2017
FAI UTB ve Zlíně

Klíčová slova: *přístup síť RADIUS SQL EDUROAM*
Anotace: *Tato práce se zabývá analýzou přístupů na sít' VŠB-TU, což je rozsáhlá počítačová sít' poskytující služby tisícům uživatelů, stanicím a mobilních zařízení. Do analýzy spadá identifikace klíčových údajů, jejich získání, ukládání a prezentace koncovému uživateli. Dále by měl systém umožnit snadnou a relativně účinnou blokaci uživatelů. Systém budou denně používat sít'oví administrátoři na VŠB-TU. Jejich potřeba disponovat takovýmto systémem vychází zejména z časové náročnosti dohledávání záznamů při běžné práci, podpoře uživatelů, ale i jejich blokací. Čas strávený nad dohledáním identity připojeného zařízení se dnes pohybuje až okolo patnácti minut a nový systém by měl umožnit tuto dobu zkrátit ideálně pod jednu minutu.*

Obsah

1. Úvod

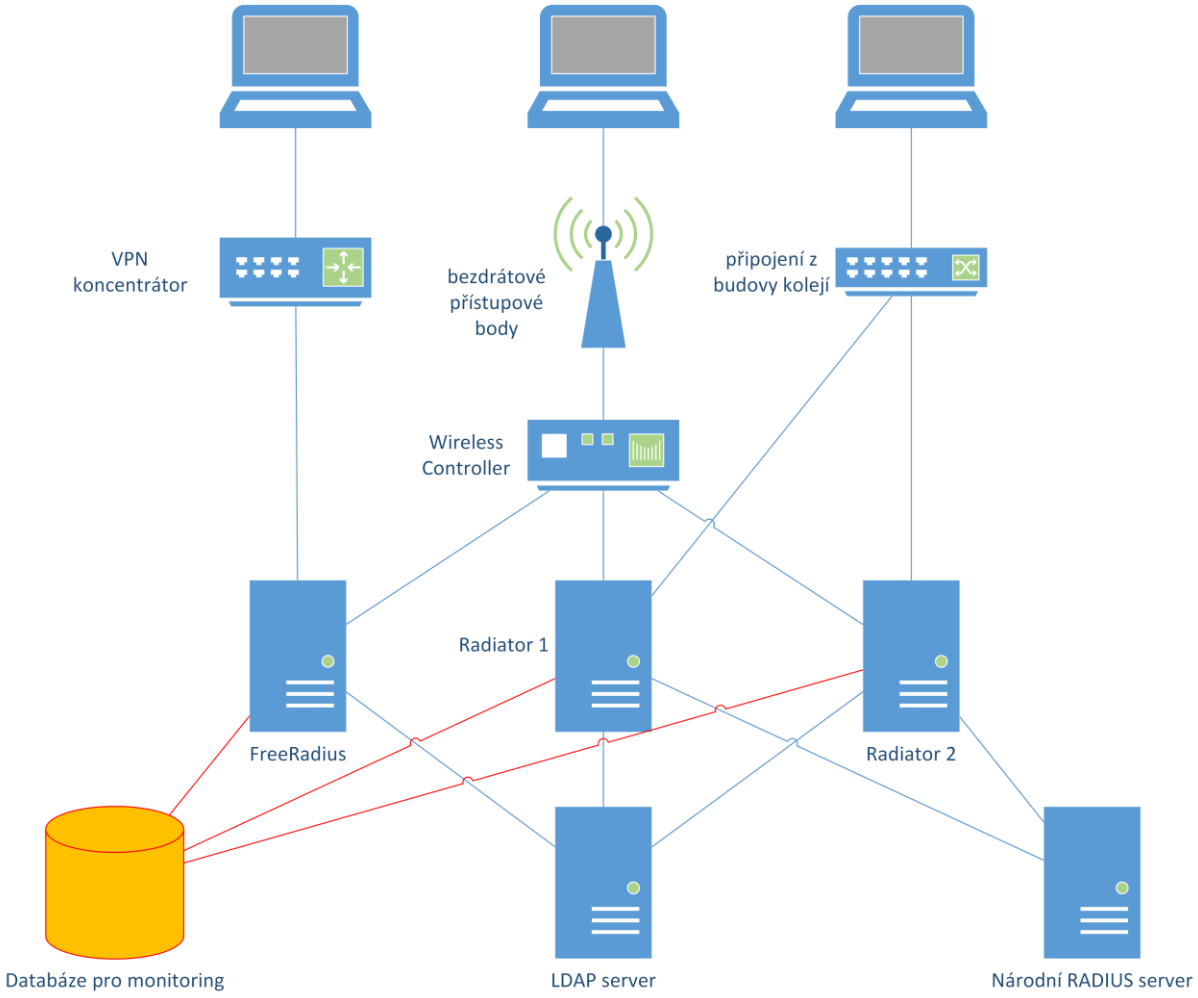
Cílem práce bylo vyvinout a popsat informační systém, který budou denně používat síťoví administrátoři na VŠB-TU. Jejich potřeba disponovat takovýmto systémem vychází zejména z časové náročnosti dohledávání záznamů při běžné práci, podpoře uživatelů, ale i jejich blokací. Čas strávený nad dohledáním identity připojeného zařízení se dnes pohybuje až okolo patnácti minut a nový systém by měl umožnit tuto dobu zkrátit ideálně pod jednu minutu. Dále by měl systém umožnit snadnou analýzu přístupů do sítě, která je s aktuálními programovými prostředky velmi pracná. Důvod, proč je vůbec potřeba tyto informace shromažďovat a vyhledávat v nich, je zejména kvůli zpřehlednění a zrychlení provozní podpory uživatelů a také kvůli zrychlení řešení bezpečnostních incidentů. Všichni uživatelé resp. jejich zařízení se do univerzitní sítě přihlašují pomocí svého jedinečného osobního jména a hesla. Tyto přihlašovací údaje ověřují přístupová zařízení (AP, přepínače) za pomoci RADIUS serveru, proto první kapitola bude věnována popisu RADIUS protokolu a to v rozsahu nutném pro pochopení a realizaci cílů této práce. Další kapitoly práce budou věnovany samotnému vývoji a implementaci programových nástrojů.

2. RADIUS server

RADIUS (z anglického Remote Authentication Dial-In User Service) je síťový protokol, umožňující centralizovanou autentizaci, autorizaci a účtování (AAA model) [2]. RADIUS protokol bývá nasazován ve větších sítích kvůli možnosti přidělit každému uživateli jedinečné přihlašovací údaje a centrálně je uchovávat. Tento princip je výhodnější než zabezpečení přístupů do sítě pomocí předsdílené fráze (z anglického pre-shared-key). Rizikem tohoto zabezpečení je, že umožňuje připojení do sítě komukoli se znalostí hesla. Taková slabina je pak zneužitelná potenciálním útočníkem k útokům na vnitřní/vnější síť se zachováním vysoké míry anonymity.

Serverem rozumíme počítač, poskytující služby klientům (model klient-server) [6]. Na univerzitní síti jsou k tomuto účelu používány programy Radiator a Freeradius. Radiator ověřuje uživatele EDUROAM, Freeradius ověřuje uživatele připojující se pomocí VPN klienta a uživatele s dočasnými účty (hosté univerzity v rámci konaných akcí). Klient, který s RADIUS serverem komunikuje, je nazýván přístupový server známý pod zkratkou NAS (z anglického Network Access Server) [2]. Jedná se většinou o síťový přepínač, Wi-Fi access-point (AP) nebo VPN koncentrátor.

Jeden z pilířů AAA modelu je tzv. účtování (z anglického accounting). Účtování je pro samotnou analýzu přístupů ten nejcennější zdroj informací. Po úspěšné autentizaci NAS kromě autorizace započne uživatelskou relaci a v průběhu této relace NAS na RADIUS server odesílá tzv. accounting pakety. Do těchto paketů se přikládají atributy, což jsou data ve tvaru název-hodnota. Nejpoužívanější atributy jsou standartizovány v RFC 2865 [4], k těmto základním atributům se však, je-li to třeba, může přidat libovolné množství atributů nestandartizovaných.



Obr.1 Nasazení RADIUS serverů na síti VŠB-TUO

3. Protokol IPv6

Ačkoli by se mohlo zdát, že již víme, kde hledat všechna pro nás zajímavá data, tak v účtování jeden velmi důležitý údaj chybí a to IPv6 adresa. IPv6 je moderní protokol používaný současně s protokolem IPv4. IPv6 je v některých ohledech od IPv4 velmi odlišný i když oba vznikly za cílem umožnit počítačům v síti komunikovat. IPv6 se dnes nasazuje převážně kvůli zaplnění adresního prostoru IPv4. Bezstavová konfigurace je nový způsob získávání komunikačních parametrů při připojení zařízení do sítě [5]. Narozdíl od stavové konfigurace, kdy veškeré komunikační parametry přiděluje DHCP server nebo se zadávají manuálně (a tudíž lze vše relativně jednoduše dohledat), tak u bezstavové konfigurace přechází část procesu na samotné koncové zařízení. V IPv6 síti s povolenou bezstavovou konfigurací si část síťových parametrů přidělí zařízení samo na základě ohlášení směrovače. Informace, které se v ohlášení směrovače nacházejí se liší podle konfigurace. Při bezstavové konfiguraci si však vždy výslednou IPv6 adresu určí zařízení. Konečnou IPv6 adresu nejsou administrátoři schopni žádným způsobem ovlivnit, proto se tato IPv6 adresa nemůže nacházet nikde v záznamech serverů, pomocí kterých by administrátoři mohli toto zařízení identifikovat. Tento problém jsem vyřešil tím, že i když nemám žádnou informaci o využitých IPv6 adresách v mé síti, tak pořád veškerý provoz probíhá přes univerzitní síťové prvky, bez kterých by komunikace uživatele nebyla možná. Nabízí se například aktivní dotazování směrovače na jeho tabulku sousedů (neighbour cache), obdobu ARP tabulky, kterou používají zařízení komunikující pomocí protokolu IPv4.

4. Volba software a konfigurace serverů

Jeden z provozních požadavků na tento informační systém je, aby běžel na Linuxovém serveru. Byla zvolena distribuce Debian, která se vyznačuje velmi vysokou spolehlivostí a stabilitou. Jako databázový server jsem zvolil PostgreSQL. Tento databázový server jsem zvolil především kvůli datovým typům - PostgreSQL obsahuje datové typy pro MAC adresu a IPv4/IPv6 adresu. Prezentaci dat administrátorům bude zajišťovat webový server Apache. IPv6 adresy uživatelů jsou získávány ze směrovačů pomocí SNMP protokolu, kde se požadovaná data nacházejí v tabulce sousedů. Pomocí tohoto způsobu ale nelze mít všechna data okamžitě a musí být zvolen interval, kdy se data budou ze směrovačů získávat. Tento interval musí být dostatečně krátký na to, aby jsme byli schopni archivovat všechny adresy, které se na síti vyskytly a zároveň dostatečně dlouhý, abychom příliš nezatěžovali směrovače a databázi. Nevýhoda tohoto řešení je, že se tabulky stahují vždy celé a v databázi bude velké množství duplicitních dat, které bude nutné pravidelně agregovat.

4.1 Konfigurace serveru RADIATOR

Server Radiator byl navržen jako modulární program [3] Jedna z jeho konfiguračních možností umožňuje při zpracovávání požadavků spustit skript napsaný v jazyce Perl, kterým můžeme upravit chování celého procesu. Protože popis celého konfiguračního souboru nespadá do rozsahu této práce, budu popisovat jen nejdůležitější parametry pro to, aby úpravy byly na již fungujícím serveru replikovatelné. Pro řešení zadaných cílů se je třeba zaměřit na tzv. klauzuli handler, která specifikuje, jaké operace se mají provést pro určité skupiny připojujících se uživatelů. Pro lepší představu uvádím část konfigurace sloužící k odbavení uživatelů VŠB.

```
<Handler Realm=/^vsb\.cz$/ix >
        AuthBy LDAP_VSB
        AuthLog vsbusers
        AcctLogFileName /var/log/radiator/radiator-detail
        WtmpFileName /var/log/radiator/wtmp
        RejectHasReason
</Handler>
```

V uvedené části konfiguračního souboru můžeme vidět, že pro uživatele s realmem vsb.cz bude provedeno ověření hesla pomocí LDAP (AuthBy), umístění textových souborů pro log (AuthLog pro autentizaci,AcctLogFileName pro účtování a WtmpFileName pro Access-Request pakety) a že v případě selhání autentizace se uživateli odešle důvod [3] lze nezdaru (RejectHasReason). Dle dokumentace [3] do konfigurace přidat návěstí AuthBy INTERNAL, kde můžeme ovlivnit vytváření odpovědi na příchozí účtovací pakety. V našem případě odpověď' měnit nechceme, můžeme ale využít toho, že v této fázi můžeme spustit námi napsaný skript, kterému server předá všechny účtovací atributy ve formě asociativního pole. Poté již není problém pomocí skriptu všechna potřebná data odeslat do databázového serveru. Pro implementaci blokace jsem nakonec použil klauzuli <AuthBy FILE>, která je primárně určená k autentizaci uživatelů jejichž údaje jsou uloženy v textovém souboru. Výsledek autentizace lze poté negovat pomocí konfiguračního parametru Blacklist. Samotná konfigurace může vypadat například takhle:

```
<AuthBy FILE>
        Identifier check_users
        NoCheckPassword
        NoEAP
        NoDefault
        Blacklist
        Filename /etc/radiator/utils/blacklistUsers.txt
</AuthBy>
```

Pro blokaci MAC adres stačí vytvořit stejnou sekci s jedinou změnou a to přidáním parametru AuthenticateAttribute Calling-Station-Id. Ten nám umožní kontrolovat místo uživatelských jmen MAC adresy. Výsledná konfigurace tedy může vypadat následovně:

```
<Handler Realm=/^vsb\.cz$/ix >
        <AuthBy INTERNAL>
                AcctHook file:"/etc/radiator/utils/sqlradacct.pl"
        </AuthBy>
        AuthByPolicy  ContinueUntilReject
        AuthBy              check_users
        AuthBy              check_mac
        AuthBy              LDAP_VSB
        AuthLog             vsbusers
        AcctLogFileName     /var/log/radiator/radiator-detail
        WtmpFileName        /var/log/radiator/wtmp
        RejectHasReason
</Handler>
```

4.2 Konfigurace serveru FreeRadius

Server FreeRadius narozdíl od serveru Radiator neumožňuje takovou flexibilitu, kde lze upravovat chování za pomocí vlastních skriptů, lze však docílit stejného chování jako v předchozím případě za použití vhodné konfigurace. Pro spolupráci FreeRadius serveru s

databází je třeba ke standartní instalaci doinstalovat modul *freeradius-postgresql* [1]. Po dokončení instalace je třeba v konfiguračním souboru radiusd.conf odkomentovat řádek *$INCLUDE sql.conf*, čímž podporu SQL povolíme. V souboru sql.conf se specifikují údaje nutné pro připojení k databázi. Nejdůležitější řádky jsou zejména tyto:

```
database      = "postgresql" #Název databázového programu
server = "sixmon.vsb.cz" #Hostname
login   = "radius" #Uživatelské jméno
password      = "heslo" #Heslo
radius_db     = "radiusdb" #Název databáze
acct_table1 = "radacct" #Název tabulky pro účtování
$INCLUDE dialup.conf #Konfigurační soubor s SQL dotazy
```

V konfiguračním souboru *./sites-enabled/default* v sekci *accounting* odkomentováním řádku sql určíme, že komunikace s databázovým serverem má probíhat pro účtování. Dále je třeba upravit konfigurační soubor *dialup.conf* kde specifikujeme jak mají dotazy odesílané na SQL server vypadat. Pro každou událost se zde konfiguruje primární a alternativní dotaz pro případ, že primární selže. Tato vlastnost mi umožnila vypořádat se s problémem, že přístupové servery do stejného atributu vkládají odlišné hodnoty. V našem případě máme v atributu *Calling-Station-Id* MAC adresu nebo IPv4 adresu, protože FreeRadius odbavuje klienty sítě tuonet-guest a zároveň klienty připojující se přes VPN. Na databázovém serveru jsem v tabulce pro účtování určil, že v sloupci *CallingStationId* do kterého se tato hodnota ukládá je datového typu *macaddr* (datový typ pro MAC adresu). Když tedy FreeRadius na databázový server odešle dotaz, kde se snaží do sloupce *CallingStationId* vložit IPv4 adresu, dotaz je odmítnut a FreeRadius odešle alternativní dotaz, kde je hodnota atributu vkládána již správně do sloupce *externalip*.

K docílení blokace uživatelů je třeba vytvořit nový konfigurační soubor, kde specifikujeme údaje pro připojení k databázovému serveru. Proto stačí zkopírovat již funkční soubor *sql.conf*, uložit ho pod jiným názvem (zvolil jsem název *sql-blacklist.conf*) a pouze upravit hodnoty jednotlivých konfiguračních parametrů. Tento soubor se zahrne do hlavního konfiguračního souboru *radiusd.conf* pomocí řádku *$INCLUDE sql-blacklist.conf*. Pro samotnou blokaci se v souboru *sites-enabled/default* do sekce *post-auth* přidají tyto řádky:

```
#Blokace uživatelského jména
if ( "%{tolower:%{User-Name}}" == "%{sql-blacklist:
        SELECT userName FROM blacklist WHERE
        username=lower('%{User-Name}')}")
        {
                reject
                update reply {
                        Reply-Message = "Your account has been blocked."
                }
}
#Blokace MAC adresy
if ( "%{tolower:%{User-Name}}" == "%{sql-blacklist:
        SELECT mac FROM blacklist WHERE
        mac=lower('%{Calling-Station-Id}')}")
        {
                reject
                update reply {
                        Reply-Message = "Your account has been blocked."
                }
```

```
}
```

4.3 Návrh SQL databáze

Kvůli rozdílnosti získávaných dat jsou vytvořeny 3 databáze, které jsou používány nezávisle na sobě. Při návrhu databáze bylo třeba zajistit, aby v tabulkách nebylo velké množství dat. Pokud bysme nechali všechna data v jedné tabulce, tak by jsme časem zaznamenali obrovskou ztrátu výkonu na databázovém serveru. Trvání jednotlivých operací by se rostoucím počtem dat postupně prodlužovalo z milisekund až na desítky sekund. Jako řešení tohoto problému jsem zvolil tzv. partitioning tabulek, což znamená, že se tabulka rozdělí na více menších tabulek podle určitého pravidla. V našem případě je pro každý den vytvořena nová tabulka. Podmínka pro existenci záznamu v téhle tabulce je, že se datum pořízení záznamu musí shodovat s datem, pro které je tabulka určena. Toto rozdělení je výhodné zejména kvůli tomu, že při vyhledávání dat administrátorem, kdy je na server odeslán příkaz SELECT, nebudou prohledávány všechny tabulky, ale jen ty, kde se záznamy opravdu mohou nacházet. Protože se záznamy neuchovávají navždy, ale po určité době se mažou, tak je možné mazat rovnou celé tabulky. V opačném případě by se musel kontrolovat záznam po záznamu, což by mělo za následek další ztrátu výkonu.

Databáze určená pro účtování obsahuje jednu tabulku určenou pro neukončené účtování (denní tabulka) a druhou tabulku pro archivaci. Nad denní tabulkou se nejčastěji provádějí příkazy INSERT pro vkládání získaných dat a příkazy SELECT, kdy se pomocí *Accounting-Update* paketů kontroluje, zda při začátku účtování v atributech nechyběla přiřazená IPv4 adresa. Protože při kontrole známe identifikátor sezení (*AcctSessionId*), můžeme záznam hledat podle tohoto identifikátoru, který je jako jediný v denní tabulce indexován. Kvůli zajištění co nejmenšího počtu řádků v denní tabulce se již ukončené relace v noci přesouvají do historické tabulky. Protože ne všechny relace jsou v době agregace ukončené musí se denní tabulka kontrolovat řádek po řádku. Kontroluje se zejména to, že řádek obsahuje nenulové hodnoty pro sloupce *AcctStartTime* a *AcctStopTime*. Takovýto záznam skript vloží do historické tabulky a z denní smaže. Agregaci zajišťuje skript napsaný v programovacím jazyce Perl. Protože je po celou dobu využíváno je pro agregaci několika tisíc záznámů používáno jen dvou dotazů, je pro dosažení větší rychlosti agregace využíváno předpřipravených dotazů. Při použití předpřipraveného dotazu se na databázový server dotaz pošle pouze jednou a poté se odesílají pouze data, která si server do dotazu dosazuje. Server tak nemusí neustále dokola parsovat a připravovat jeden a ten samý dotaz [7].

název	datový typ	index	index - archív
radacctid	bigserial	ano	ano
AcctSessionId	text	ano	ano
UserName	text	ne	ano
FramedIPAddress	inet	ne	ano
CallingStationId	macaddr	ne	ano
AcctStartTime	timestamp with timezone	ne	ano
AcctStopTime	timestamp with timezone	ne	ano
externalip	inet	ne	ano
ipv6pref	cidr	ne	ne
ipv6id	text	ne	ne
framedipv6	inet	ne	ano
calledstationid	text	ne	ne
nasidentifier	text	ne	ne

tab.1 tabulka pro účtování

V součtu se v tabulkách sousedů běžně nachází několik tisíc párů MAC a IPv6 adres. Protože získávání informací o IPv6 adresách vyskytujících se na síti probíhá v pravidelných intervalech stahováním celých tabulek sousedů, bude tento proces hlavním důvodem špiček na databázovém serveru. Tabulka, do které tato data přicházejí proto musí být pokud možno co nejvíce efektivní. Z tohoto důvodu byla vytvořena tabulka určená pouze ke sběru dat. Protože jsme se chtěli vyhnout tomu, aby se v tabulce u jednotlivých párů aktualizoval čas, kdy byly v síti spatřeny, nad touto tabulkou se při sběru informací spouští pouze příkaz INSERT. Indexace nad touhle tabulkou neprobíhá, což příkaz INSERT zrychlí.

název	datový typ	index
sweep_time	time	ne
ipv6	inet	ne
mac	macaddr	ne
date	date	ne

tab.2 Tabulka pro uchovávání IPv6 adres

název	datový typ	index
id	bigserial	ano
ipv6	inet	ano
mac	macaddr	ano
first_seen	time	ne
last_seen	time	ne
date	date	ne

tab.3 Agregovaná tabulka - IPv6 adresy

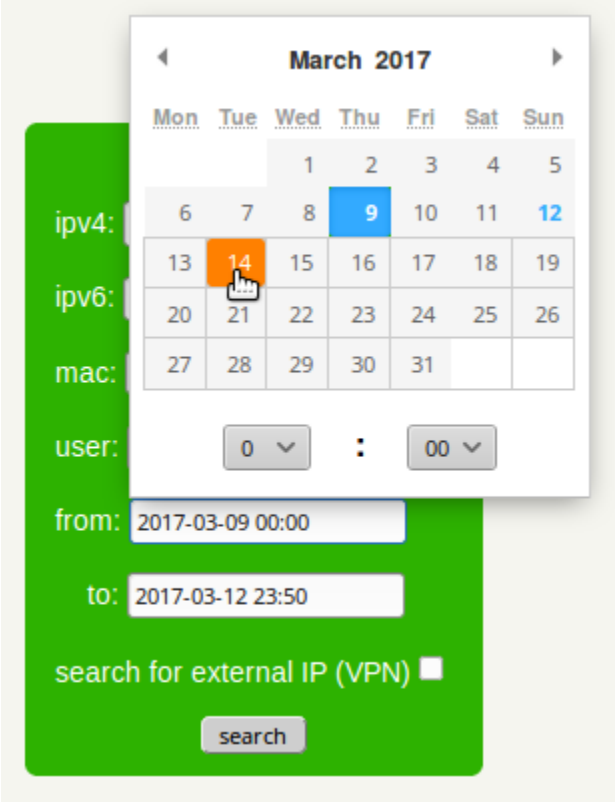
Při první implementaci funkcionality pro blokování uživatelů na síti byla vytvořena jednoduchá tabulka se sloupci pro MAC adresu a pro login uživatele. Pro lepší orientaci v datech byly přidány ještě další sloupce, které se na funkci nepodílejí, ale administrátorům poskytují dodatečné informace jako je čas blokace, kdo blokaci provedl a důvod blokace. Po provedení prvních blokací jsem byl nucen logiku blokací dodatečně upravit z důvodu, že někteří uživatelé blokace obcházeli tím, že přesvědčili své spolužáky, aby jim dali své přihlašovací údaje. Webové rozhraní proto při blokaci loginu uživatele projde databázi pro účtování a nalezne všechny MAC adresy pod kterými se kdy uživatel připojil. Tyhle MAC adresy jsou poté blokovány taky a jsou od ostatních odlišeny pomocí sloupce *owner*. Díky tomu můžou být při odblokaci snadno nalezeny a z tabulky smazány. Vedlejší efekt tohoto přístupu je, že můžou být zablokována i zařízení, která blokovaný uživatel nevlastní. Systém na tento případ administrátora upozorní a je na něm, zda takové zařízení do blokace zahrne.

4.4 Webové rozhraní

Webové rozhraní slouží jako jednoduchá cesta, jak se samotným systémem pracovat. Obsluha se tedy nebude muset připojovat přímo k databázi a žádaná data získávat pomocí ručně psaných SQL dotazů. Výsledky vyhledávání jsou navíc obohacené o reverzní DNS překlady IPv4 adres a výrobce síťových karet. Obsluha může vyhledávat účtovací logy a blokované uživatele. Mezi kategoriemi vyhledávání se přepíná pomocí postranního menu. Webové rozhraní navíc umožňuje kromě autentizace i autorizaci uživatele.

Vyhledávání v logu účtování využijí především členové bezpečnostního týmu při dohledávání identity zařízení, které figurovalo v bezpečnostním incidentu. V praxi se však v logu hledalo i kvůli provozní podpoře uživatelů. Často se totiž stává, že uživatel, který se kvůli problému obrátí na helpdeskové pracoviště neoznámí, že byl jeho problém vyřešen a operátor helpdesku neví, jestli může požadavek uzavřít. Dále se dá pomocí tohoto logu velmi rychle vyloučit problém s autentizací na síti a lze se zaměřit na hledání problémů jiných.

K vyhledávání v logu slouží jednoduchý formulář, kde lze zadat více vyhledávacích kritérií. Nalezeny budou záznamy, které obsahují právě všechna zadaná kritéria. Pro provedení vyhledávání musí být vyplněno alespoň jedno kritérium a vymezená časová oblast, kde se má záznam přibližně nacházet. Kdyby nebyla zadána časová oblast, nemohlo by se využít partitioningu tabulek v SQL databázi a tím k nadměrné zátěži. Tato podmínka je také důležitá kvůli tomu, že prohlížeče kolabují, když dojde k výpisu velkého množství záznamů (řádově tisíců). Pro snadnější zadávání správného formátu data do formuláře byly do rozhraní implementovány projekty *Momentum* a *Pikaday*, kdy si obsluha datum navolí pomocí kalendáře.



obr.2 vyhledávací formulář

obr.3 blokace uživatele

4.5 Závěr

V práci byly teoreticky popsány nejdůležitější aspekty ohledně RADIUS protokolu, které jsou klíčové pro vývoj a implementaci požadovaných funkcionalit. Díky těmto poznatkům jsem vyvinul programové moduly upravující chování programu Radiator a ty úspěšně implementoval do produkčního prostředí. Ke správné funkci modulů bylo třeba také navrhnout databázi s ohledem na co nejvyšší rychlost zpracování dotazů, aby nedocházelo ke zpožd'ování operací probíhajících na RADIUS serverech. K zajištění této vlastnosti jsem vytvořil agregační skripty pro oddělení aktuálních a historických dat. Pro zajištění větší rychlosti vyhledávání historických dat byla použita technika zvaná partitioning. Kvůli absenci informací o IPv6 adresách jsem také naprogramoval skripty, které pravidelně stahují tabulky sousedů ze směrovačů. Dále jsem naprogramoval webové rozhraní. Webové rozhraní přehledně zobrazuje uložené údaje a umožňuje autentizaci a autorizaci pověřených pracovníků.

Implementace těchto nástrojů zahrnovala úpravu konfigurace produkčních RADIUS serverů a úplnou konfiguraci nově nainstalovaného SQL serveru a webového serveru. Uvedené nástroje již byly otestovány a jsou nasazeny v běžném provozu.

Literatura

[1] Guide/SQL HOWTO 2016. The FreeRADIUS Server Project and Contributors [online]. 2016 [cit. 2017-01-11]. Dostupné z: https://wiki.freeradius.org/guide/SQL-HOWTO

[2] HASSELL, Jonathan 2002. RADIUS. 1005 Gravenstein Highway North, Sebastopol, CA95472: O'Reilly Media, 2002. ISBN 0-596-00322-6.

[3] RADIUS Server 2015 [online]. Open System Consultants Pty., 2015 [cit.Radiator 2016-12-30]. Dostupné z: https://www.open.com.au/radiator/ref.pdf

[4] Remote Authentication Dial In User Service (RADIUS) 2000. The Internet Engineering Task Force [online]. 2000 [cit. 2016-11-27]. Dostupné z: https://tools.ietf.org/html/rfc2865

[5] SATRAPA, Pavel 2011. IPv6: internetový protokol verze 6. 3., aktualiz. a dopl. vyd. Praha: CZ.NIC, c2011. CZ.NIC. ISBN 9788090424845.

[6] Server Definition 2014. TechTerms [online]. 2014 [cit. 2016-11-27]. Dostupné z: http://techterms.com/definition/server

[7] ŽÁK, Karel 2004. PostgreSQL: připravené dotazy a oddělení dat od dotazů. Root.cz [online]. 2004 [cit. 2016-12-30]. Dostupné z: https://www.root.cz/clanky/postgresql-pripravene-dotazy-a-oddeleni-dat-od-dotaz