МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Керування корпоративним ДНС сервером»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Ободяк В.К..**

Студентки групи ІНдн – 61с — **Дмітрієва В.Б.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Центр заочної, дистанційної і вечірньої форм навчання

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ
до випускної роботи

Студентки четвертого курсу, групи ІНдн-61С спеціальності "Інформатика" заочної форми навчання Дмітрієвої Владислави Богданівни.

Тема: "Керування корпоративним сервером ДНС "

Затверджена наказом по СумДУ

№_____ от _____2020 р.

Зміст пояснювальної записки: 1) аналітичний огляд теорії ДНС; 2) постановка завдання й формування завдань дослідження; 3) опис основних положень, конфігурацій що використовуються технологією ДНС; 4) розробка програмного забезпечення автоматизації ДНС.

Дата видачі завдання "_____"__ _____2020 р.

Керівник випускної роботи _____ Ободяк В.К..

Завдання приняла до виконання _____ Дмітрієва В.Б.

РЕФЕРАТ

Записка: 77 стор., 14 рис., 1 додаток, 17 джерел.

Об’єкт дослідження — процесс автоматизації управління корпоративним сервером ДНС.

Мета роботи — розробка програмного забезпечення для автоматизації процесів керування корпоративним ДНС сервером.

Методи дослідження — використання програмних скриптів для автоматизації управління корпоративним сервером ДНС.

Результати — розроблено рекомендації та програмне забезпечення щодо автоматизації керування корпоративними доменними зонами. Розроблений код реалізовано у формі програмного забезпечення, створеного за допомогою мови програмування PHP.

АВТОМАТИЗАЦІЯ КОРПОРАТИВНОГО ДНС, ДНС СЕРВЕР, АВТОМАТИЗАЦІЯ СЕРВІСУ, КЕРУВАННЯ ДОМЕНИМИ ЗОНАМИ

Зміст

ВСТУП 5

1. АНАЛІЗ ПРОБЛЕМИ, ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ 6

1.1 Аналіз існуючого ДНС сервера СумДУ 6

1.2 Проблема і постановка задачі 8

2. ТЕОРЕТИЧНА СКЛАДОВА ДНС 9

2.1 Історія розвитку ДНС 9

2.2 Принципи організації DSN 12

2.3 Типи записів ДНС 42

3. РЕАЛІЗАЦІЯ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ДНС СЕРВЕРОМ. 59

3.1. Створення БД проекту 59

3.2. Створення конфігураційного файлу Apache2 60

3.3. Створення скриптів автоматизації управління. 62

ВИСНОВКИ 74

СПИСОК ЛІТЕРАТУРИ 75

ПЕРЕЛІК СКОРОЧЕНЬ 77

ДОДАТОК. 78

ВСТУП

Сучасний світ корпоративного сегмента інтернет вже давно переріс грань маленької контори з 15 комп'ютерами та універсальними системними адміністраторами, які вміють все і відразу. Сьогодні ринок IT послуг - це величезний список, що включає в себе більше 100 найменувань продуктів і послуг, в кожному з яких залучаються сотні різних фахівців.

Величезні компанії, які використовують у своїй діяльності широку структуру веб ресурсів і мають в наявності досить великий парк обладнання, не можуть проводити свою діяльність в інтернет без наявності корпоративного ДНС сервера.

Управління власним доменним ім'ям: швидкий контроль над ним, легкість і динамічність його управління - це важливі характеристики успішної компанії в інтернет.

Використовуючи сучасні технології, організації можуть швидко і якісно відповідати на всі вимоги сучасних політик інтернет. Швидке створення нового сайту багато в чому залежить від якості і швидкості роботи ДНС сервера підприємства і його відмовостійкості.

У даній роботі розглянуто процес налаштування і експлуатації корпоративного ДНС сервера на прикладі організації Сумського державного університету.

1. АНАЛІЗ ПРОБЛЕМИ, ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз існуючого ДНС сервера СумДУ

На сьогодні сервер управління ДНС в СумДУ управляється за допомогою ручного керування доменною зоною, за допомогою прописування доменних записів вручну в текстовому файлі конфігурації named.

Сервер управління зоною працює на 1 зовнішній адресі разом з динамічним проксі-сервером доступу в мережу Інтернет всіх користувачів локальної мережі університету.

Доменні зони, що знаходяться в зоні контролю сервера:

Sumdu.edu.ua:

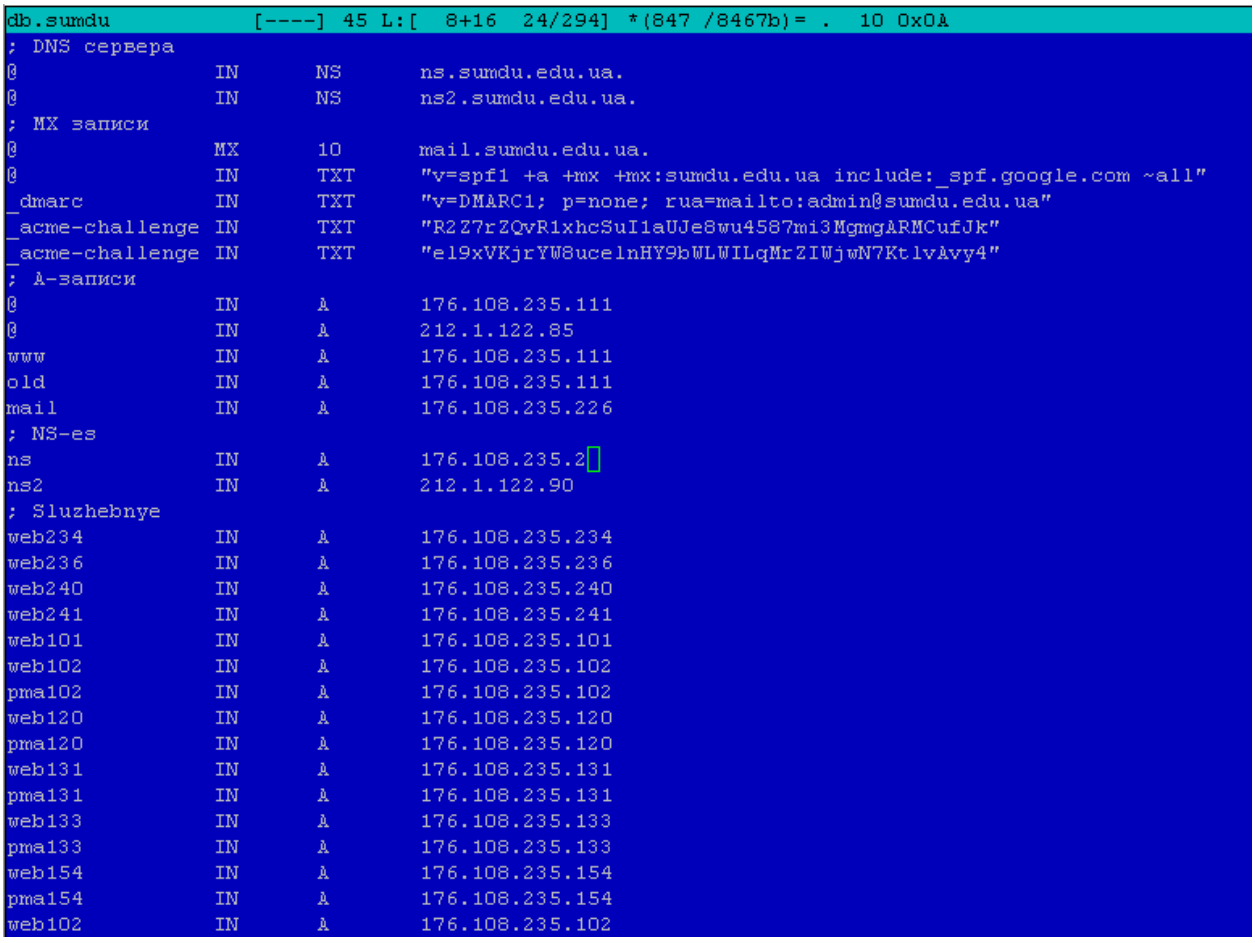
Sudu.edu.ua:

Uabs. edu. ua.

Алгоритм дій адміністратора ДНС такий:

1. Адміністратор здійснює ssh - тунельне з'єднання з сервером ДНС;
2. За допомогою вбудованого текстового реактора вносяться зміни в файлі конфігурації відповідної доменної зони;
3. Перезавантаження служби named.

Приклад конфігураційного файлу наведено в рис. 1.1.



```
db.sumdu            [----] 45 L:[  8+16  24/294] *(847 /8467b)= .  10 0x0A
; DNS сервера
@                   IN      NS      ns.sumdu.edu.ua.
@                   IN      NS      ns2.sumdu.edu.ua.
; MX записи
@                   MX      10      mail.sumdu.edu.ua.
@                   IN      TXT     "v=spf1 +a +mx +mx:sumdu.edu.ua include:_spf.google.com ~all"
_dmarc              IN      TXT     "v=DMARC1; p=none; rua=mailto:admin@sumdu.edu.ua"
_acme-challenge     IN      TXT     "R2Z7rZQvR1xhcSuI1aUJe8wu4587mi3MgmgARMCufJk"
_acme-challenge     IN      TXT     "el9xVKjrYW8ucelnHY9bWLWILqMrZIWjwN7KtlvAvy4"
; A-записи
@                   IN      A       176.108.235.111
@                   IN      A       212.1.122.85
www                 IN      A       176.108.235.111
old                 IN      A       176.108.235.111
mail                IN      A       176.108.235.226
; NS-es
ns                  IN      A       176.108.235.2
ns2                 IN      A       212.1.122.90
; Sluzhebnye
web234              IN      A       176.108.235.234
web236              IN      A       176.108.235.236
web240              IN      A       176.108.235.240
web241              IN      A       176.108.235.241
web101              IN      A       176.108.235.101
web102              IN      A       176.108.235.102
pma102              IN      A       176.108.235.102
web120              IN      A       176.108.235.120
pma120              IN      A       176.108.235.120
web131              IN      A       176.108.235.131
pma131              IN      A       176.108.235.131
web133              IN      A       176.108.235.133
pma133              IN      A       176.108.235.133
web154              IN      A       176.108.235.154
pma154              IN      A       176.108.235.154
web102              IN      A       176.108.235.102
```

Рисунок 1.1 - Приклад існуючого конфігураційного файлу

1.2 Проблема і постановка задачі

В існуючій конфігурації в зіставленні зі світовими стандартами були виявлені такі недоліки:

1. Управління доменними зонами відбувається в режимі ручного зміни конфігураційного файлу.
2. ДНС сервер - фізично один пристрій.

Такі недоліки можуть призвести до часткової або повної втрати контролю над доменним ім'ям в зв'язку з декількома факторами, такими як людський фактор або фактор вимкнення сервера. У випадку з людським фактором не виключені помилки в роботі адміністратора з наступною відсутністю працездатності сервера, що може привести до тимчасової або постійної втрати контролю над ДНС сервером. Другий недолік може бути реалізований за допомогою проблем з наданням послуги інтернет провайдером або відсутністю подачі струму безпосередньо на сервері ДНС, що в свою чергу може призвести до тимчасової втрати контролю над доменними записами.

Аналіз існуючого ДНС сервера СумДУ показав що для вирішення проблеми керування корпоративним ДНС сервером, потрібно вирішити такі задачі:

1. Проаналізувати технологію обробки даних та типи записів ДНС.
2. Проаналізоувати сучасний стан роботи ДНС в Сумському державному університеті.
3. Розробити прототип ДНС серверу університету.
4. Розробити процес автоматизації управління ДНС сервером.

2. ТЕОРЕТИЧНА СКЛАДОВА ДНС

2.1 Історія розвитку ДНС

Свій початок система доменних імен бере в 50-х - 60-х роках минулого століття. Тоді вона допомогла спростити адресацію хостів в мережі ARPANET і дуже швидко перейшла від обслуговування сотень комп'ютерів до роботи з сотнями мільйонів. З чого починалася DNS ?

У 1958 році уряд США заснувало Агентство передових дослідницьких проектів (ARPA). Зусилля організації направили на розробку технологій у сфері зберігання та передачі даних. У 60-х роках агентство отримало нове апаратне забезпечення - комп'ютер Q-32 - одну з найбільших обчислювальних систем на транзисторах вагою понад 60 тон. Вона мала відразу два запам'ятовуючих пристрої на магнітних барабанах, кожне з яких прочитував і записували по 50 біт інформації. У той час Q-32 використовували для вирішення завдань Міністерства оборони США.

Тоді дані між обчислювальними системами переносили за допомогою перфокарт, що істотно ускладнювало і уповільнювало розрахунки. Пошук нового рішення військові довірили агентству ARPA в 1968 році. Його інженери об'єдналися з колегами з MIT і розробили протокол пакетної комутації. З його допомогою вони «пов'язали» Q-32 з університетської машиною TX-2 (саме на ній піонер інтернету Айвен Сазерленд написав Sketchpad - прародителя сучасних CAD).

Протокол вдосконалювали всю першу половину 1969 року. Фахівці працювали над рівнями взаємодії комп'ютерів в мережі: апаратним, програмним і модемним. У другій половині року провели перше випробування технології. Мережа складалася з двох терміналів, встановлених на відстані 600 км в Каліфорнійському і Стенфордському університетах. В якості терміналів виступили 16-розрядні міні-комп'ютери Honeywell DDP-316 з 12 кілобайтами оперативної пам'яті. Під час тесту перший оператор вводив слово login на

одній машині, а другий підтверджував, що бачить його на екрані інший. Експеримент пройшов вдало, поклавши початок мережі ARPANET [16].

У 1983 році інженери Пол Мокапетріс (Paul Mockapetris) і Джон Постел (Jon Postel) вирішили поширити концепцію, описану в RFC805, на всю мережу ARPANET. Вони підготували два нових RFC, в яких виклали основи DNS. У RFC882 «Domain Names: Concepts and Facilities» [10] були описані можливості системи доменних імен, а в RFC883 «Domain Names: Implementation and Specification» [11] приведена деталізація специфікації і методи впровадження.

Зокрема, Мокапетріс запропонував структуру ідентифікаторів хостів, що містить ім'я і спеціальну категорію. Через рік на основі специфікації Мокапетріс народилася класифікація gTLD (generic Top-Level Domains), куди увійшли домени.com,.edu,.net,.org,.int,.gov i.mil.

Перший час ними керувала компанія Network Solutions Inc., яку для цих цілей найняло американський уряд. Пізніше кермо влади перейшли в руки спеціально створеної некомерційної організації ICANN. У 1985 році, після впровадження DNS в ARPANET, свої домени зареєстрували відразу шість організацій. Найперший з них - Symbolics.com - існує до цих пір. Сьогодні це цифровий музей історії інтернету.

З 1985 року система доменних імен зазнала безліч змін. Наприклад, в неї додали підтримку механізмів NOTIFY і IXFR, спростити процеси реплікації баз DNS між різними серверами.

Після того як в 1983 році Пол Мокапетріс (Paul Mockapetris) і Джон Постел (Jon Postel) запропонували концепцію доменних імен для мережі ARPANET, вона досить швидко здобула схвалення IT-спільноти. Одними з перших реалізувати її на практиці взялися інженери з Університету в Берклі [17]. У 1984 році чотири студента представили перший DNS-сервер - Berkeley Internet Name Domain (BIND) [16]. Вони працювали в рамках гранту, виданого Управлінням перспективних дослідницьких проектів Міністерства оборони США (DARPA).

Розроблена учнями університету система автоматично перетворювала DNS-ім'я в IP-адресу і навпаки. Цікаво, що коли її код завантажили на BSD (систему поширення ПО), перші вихідні вже мали номер версії 4.3. Перший час DNS-сервером користувалися співробітники лабораторій університету. Аж до версії 4.8.3 за розробку BIND відповідали члени дослідницької групи Університету в Берклі - Computer Systems Research Group (CSRG), але в другій половині 1980-х DNS-сервер вирвався за межі вузу - його передали в руки Пола Викси (Paul Vixie) з корпорації DEC. Пол випустив оновлення 4.9 і 4.9.1, а потім заснував Internet Software Consortium (ISC), який з тих пір і відповідає за підтримку BIND. За словами Пола, всі попередні версії спиралися на код студентів з Берклі, і за минулі п'ятнадцять років він повністю вичерпав свої можливості для модернізації. Тому 2000 року BIND переписали з нуля [16].

Сервер BIND включає в себе відразу кілька бібліотек і компонентів, що реалізують «клієнт-серверну» архітектуру DNS і відповідають за налаштування функцій DNS-сервера. BIND широко поширений, особливо на Linux, і залишається популярною реалізацією DNS-сервера. Це рішення встановлено на серверах, що забезпечують підтримку кореневої зони.

Є й альтернативи BIND. Наприклад, PowerDNS, що йде в комплекті з Linux-дистрибутивами. Він написаний Бертом Хуберта (Bert Hubert) з голландської компанії PowerDNS.COM і підтримується open source спільнотою. У 2005 році PowerDNS впровадили на серверах фонду «Вікімедіа». Рішенням також користуються великі хмарні провайдери, європейські телекомунікаційні компанії і організації зі списку Fortune 500.

BIND і PowerDNS одні з найпоширеніших, але не єдині DNS-сервери. Також варто відзначити Unbound, djbdns і Dnsmasq.

За всю історію DNS в її специфікацію вносили безліч змін. В якості одного з перших і великих оновлень додали механізми NOTIFY і IXFR в 1996 році. Вони спростили реплікацію баз даних системи доменних імен між первинним і вторинним серверами. Нове рішення дало можливість налаштовувати повідомлення про зміну DNS-записів. Такий підхід гарантував

ідентичність вторинної та первинної DNS-зони плюс економив трафік - синхронізація відбувалася тільки при необхідності, а не через фіксовані інтервали [17].

Спочатку DNS-мережа була недоступна для широкої публіки і потенційні проблеми з ІБ не були пріоритетом при розробці системи, але такий підхід дав про себе знати згодом. З розвитком інтернету уразливості системи почали експлуатувати - наприклад, з'явилися такі атаки як DNS-спуфинг. В цьому випадку кеш DNS-серверів наповнюють даними, які не мають авторитетного джерела, і перенаправляють запити на сервери зловмисників.

Щоб вирішити проблему, в DNS впровадили кріптоподписі для DNS-відповідей (DNSSEC) - механізм, що дозволяє побудувати ланцюжок довіри для домену від кореневої зони. Відзначимо, що аналогічний механізм додали для аутентифікації хостів при передачі DNS-зони - він отримав назву TSIG.

Модифікації, що спрощують реплікацію баз DNS і виправляють проблеми безпеки, IT-ком'юніті всіляко вітало. Але були і зміни, які спільнота сприйняла не кращим чином. Зокрема, перехід від безкоштовних до платних доменних імен. І це приклад лише однієї з «війн» в історії DNS.

2.2 Принципи організації DSN

Будь-яка DNS є прикладним процесом, який працює над стеком TCP/IP. Таким чином, базовим елементом адресації є IP-адреса, а доменна адресація виконує роль сервісу. Правда про те, що DNS - це прикладна задача в повному сенсі цього слова доводиться говорити з певними застереженнями. DNS - це інформаційний сервіс Internet, і, отже, протоколи його реалізують відносяться до протоколів прикладного рівня згідно зі стандартною моделі OSI. Однак з точки зору операційної системи підтримка DNS може входити в неї як компонента ядра, яка прикладним призначеним для користувача процесом не є. Призначені для користувача програми спілкуються з нею за допомогою системних викликів. Такий стан речей справедливо практично для всіх Unix-

систем. Інша справа системи на базі MS-DOS і Windows 3.x. У цих системах DNS (точніше її клієнтська частина) реалізована як прикладна програма.

Система доменних адрес будується за ієрархічним принципом. Однак ієрархія ця є лаконічною. Фактично, немає єдиного кореня всіх доменів Internet. Якщо бути більш точним, то такий корінь в моделі DNS є. Він так і називається «ROOT». Однак, єдиного адміністрування цього кореня немає. Адміністрування починається з доменів верхнього, або першого, рівня. У 80-і роки були визначені перші домени цього рівня: gov, mil, edu, com, net. Пізніше, коли мережу переступила національні кордони США з'явилися національні домени типу: uk, jp, au, ch, і т.п.

Слідом за доменами верхнього рівня слідують домени, що визначають або регіони (sumy.ua), або організації (edu.ua). В даний час практично будь-яка організація може отримати свій власний домен другого рівня. Для цього тільки треба направити заявку провайдеру і отримати повідомлення про реєстрацію. Далі йдуть такі рівні ієрархії, які можуть бути закріплені або за невеликими організаціями, або за підрозділами великих організацій.

Всю систему доменної адресації можна представити так, як показано на рисунку 2.1 :

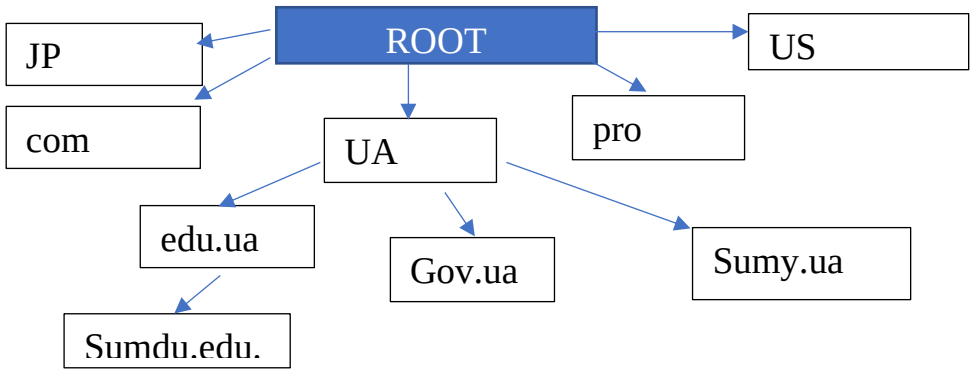


Рисунок 2.1 - Приклад ієрархії світового ДНС

Найбільш популярною програмою підтримки DNS є named, яка реалізує Berkeley Internet Name Domain (BIND). Але ця програма не єдина. Так в системі Windows NT 4.0 є свій сервер доменних імен, який підтримує специфікацію BIND. Визначено в документі RFC 1033-1035 [10,11].

BIND або **Berkeley Internet Name Domain** - це сервер доменних імен реалізований в університеті Берклі, який широко застосовується в Intenet. Він забезпечує пошук доменних імен і IP-адрес для будь-якого вузла Мережі. BIND забезпечує розсилку повідомлень електронної пошти через вузли Internet. Якщо говорити більш точно BIND забезпечує пошук доменного адреси машини користувача, якому адресована пошта, і визначення IP-адреси доменному адресою. Ця інформація використовується програмою розсилки листів sendmail, яка виступає в якості поштового сервера.

BIND реалізований за схемою «клієнт/сервер». Власне BIND - це сервер, а функції клієнта виконує name resolver або просто resolver. Зазвичай, модулі resolver 'а знаходяться в бібліотеці libc.a, якщо мова йде про систему UNIX, і редагуються разом з програмою, що використовує виклики gethostbyname і gethostbyaddr. Як уже зазначалося, базовим поняттям для BIND є «домен». На рис. 2.1 представлена схема опису системи доменів.

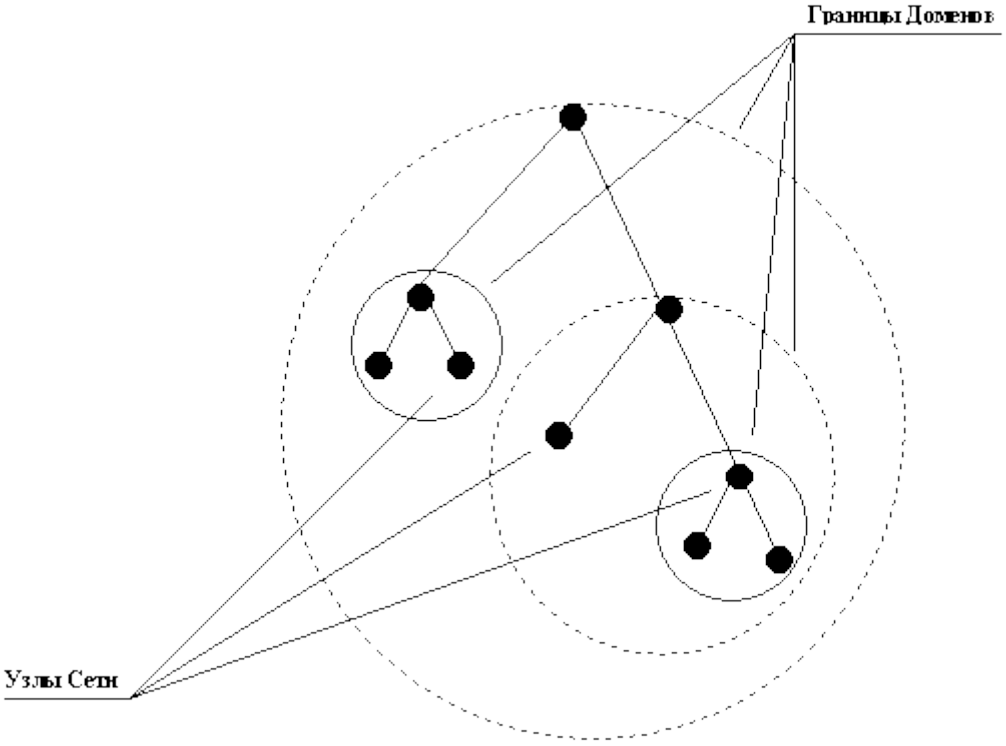


Рисунок 2.2 - Схема доменної адресації

На рисунку 2.2 домени зображені у вигляді кіл. Кожне коло відповідає безлічі комп'ютерів, які утворюють домен. Вкладені кола - це піддомени, які,

в свою чергу теж можуть бути розбиті на піддомени і т.д. Під час відображення цієї структури в доменні імена виходить ієрархія імен вузлів мережі.

Однак розмови про архітектуру сервісу доменних імен і різних його реалізацій абсолютно безпідставні, якщо тільки не мати свого власного домену. Адже проблеми з адмініструванням виникають тільки в цьому випадку. У наступному розділі розказано про те, як його отримати.

Як і будь-який інший сервіс прикладного рівня, а система доменних імен - це сервіс прикладного рівня, програма named використовує транспорт TCP і UDP (порти 53) [2]. Дуже часто користувачі повідомляють адміністратору системи, що та чи інша машина системі не відома, хоча вчора з нею можна було працювати. При цьому, як правило, називають доменні імена комп'ютерів. Перше, що слід перевірити в цій ситуації - реальну доступність до комп'ютера по його IP-адреси. Проблема дійсно є серйозною, якщо і по IP-адреси можна «достукатися» до віддаленої машини, в іншому випадку слід шукати помилки або відмови в роботі сервісу доменних імен.

Сервіс BIND будується за схемою «клієнт-сервер». Як клієнтська частина виступає процедура розпізнавання імен - resolver, а в якості сервера, в нашому випадку, програма named [6].

Resolver, власне, не є якою-небудь програмою. Це набір процедур з системної бібліотеки (libc.a), які дозволяють прикладній програмі, відредагованого з ними, отримувати по доменному імені IP-адреса комп'ютера або по IP-адреси доменне ім'я. Самі ці процедури звертаються до системної компоненті resolver, яка веде діалог з сервером доменних імен і таким чином обслуговує запити прикладних програм користувача.

На запити описаних вище функцій в системах Unix відповідає програма named. Ідея цієї програми проста - забезпечити як дозвіл, так званих, «прямих» запитів, коли в назві шукають адреса, так і «зворотних», коли за адресою шукають ім'я. Управляється named спеціальною базою даних, яка складається з декількох фалів, і містить відповідності між адресами та іменами, а також

адреси інших серверів BIND, до яких даний сервер може звертатися в процесі пошуку імені або адреси.

Загальну схему взаємодії різних компонентів BIND можна зобразити так, як це представлено на рисунку 2.3.

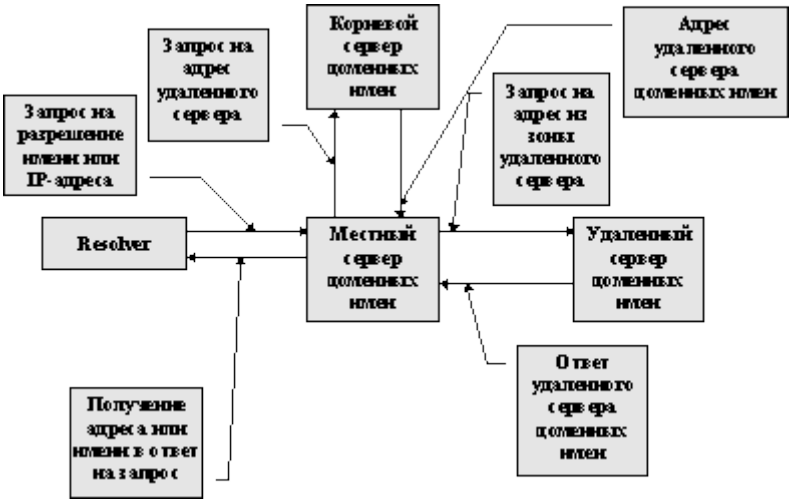


Рисунок 2.3 - Нерекурсивна процедура на дозвіл імені

Спираючись на схему нерекурсивною процедури дозволу імені (рисунок 2.3) розглянемо два способи вирішення запиту на отримання IP-адреси по доменному імені.

У першому випадку будемо розглядати запит на отримання IP-адреси в рамках зони відповідальності даного місцевого сервера імен:

1. Прикладна програма через resolver запитує IP-адреса по доменному імені у місцевого сервера.

2. Місцевий сервер повідомляє прикладній програмі IP-адреса запитаного імені.

Для того, щоб ще більше прояснити дану схему взаємодії розглянемо кілька прикладів, коли з'являється запит на отримання IP-адреси по доменному імені.

При вході в режимі віддаленого терміналу на комп'ютер vpn.sumdu.edu.ua по команді:

telnet vpn.sumdu.edu.ua

Ми отримуємо у відповідь:

telnet vpn.sumdu.edu.ua

trying 193.34.92.148...

login : …….

Рядок, в якій вказано IP-адреса комп'ютера vpn.sumdu.edu.ua, показує, що до цього часу доменне ім'я було успішно дозволено сервером доменних імен і прикладна програма, в даному випадку telnet отримала на свій запит IP-адреса. Таким чином, після введення команди з консолі і до появи IP-адреси на екрані монітора прикладна програма здійснила запит до сервера доменних імен і отримала відповідь на нього.

Досить часто можна зіткнутися з ситуацією, коли після введення команди досить довго доводиться чекати відповіді віддаленої машини, але зате після першого відповіді віддалений комп'ютер починає реагувати на команди з такою ж швидкістю, як і ваш власний. В даному випадку, швидше за все в затримці винен сервіс доменних імен. Найбільш помітно такий ефект проявляється при використанні програми ftp. Наприклад, при зверненні до фінського архіву ftp. funet.fi, після введення команди:

/home/doc > ftp ftp.funet.fi

система довго не відповідає, але потім починає швидко "ковтайте" і ідентифікатор і поштову адресу, якщо входять як анонімний користувач, і інші FTP -команди.

Інший приклад того ж сорту - це програма traceroute**.** Тут затримка на запити до сервера доменних імен проявляється в тому, що час відповідей зі шлюзів, на яких "вмирають" ICMP пакети, вказане в звіті, маленьке, а затримки з відображенням кожного рядка звіту досить великі. В системі Windows 3.1 деякі програми трасування, показують спочатку IP-адресу шлюзу, а тільки потім, після дозволу «зворотного» запиту, замінюють його на доменне ім'я шлюзу. Якщо сервіс доменних імен працює швидко, то ця підміна практично непомітна, але якщо сервіс працює повільно, то проміжки бувають досить значними.

Якщо в прикладі з telnet і ftp ми розглядали, тільки «прямі» запити до сервера доменних імен, то в прикладі з traceroute ми вперше згадали «зворотні» запити. У «прямому» запиті прикладна програма запитує у сервера доменних імен IP-адресу, повідомляючи йому доменне ім'я. При «зворотному» запиті прикладна програма запитує доменне ім'я, повідомляючи сервера доменних імен IP-адресу.

Слід зауважити, що швидкість дозволу «прямих» і «зворотних» запитів в загальному випадку різна. Все залежить від того як описані «прямі» і зворотні «зони» в базах даних серверів доменних імен, які обслуговують домен. Часто пряма зона буває одна, а ось зворотних зон може бути кілька, в силу того, що домен розташований на фізично різних мережах або підмережах. В цьому випадку час дозволу «зворотних» запитів буде більше часу дозволу «прямих» запитів.

Процедура дозволу «зворотних» запитів точно така ж, як і процедура дозволу «прямих» запитів.

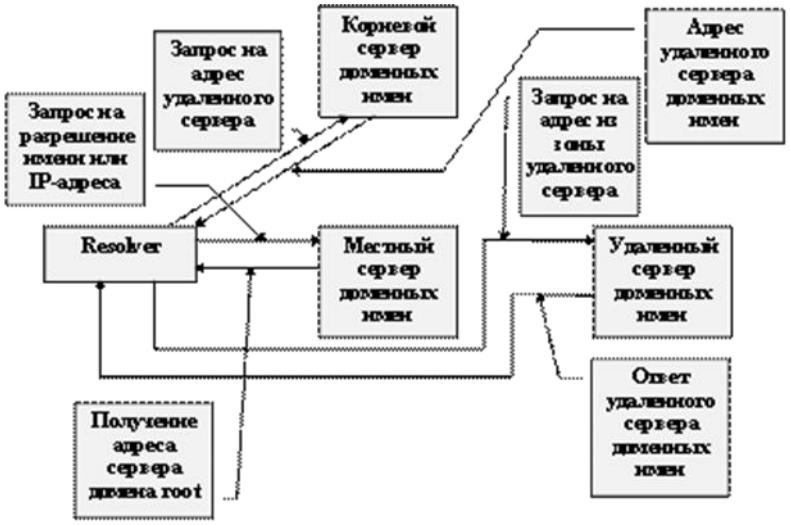


Рисунок 2.4 - Нерекурсивний запит resolver

Власне нерекурсивне, розглянутий вище запит є тільки з точки зору сервера, тобто програми named. З точки зору resolver процедура дозволу запиту є рекурсивною, так як resolver передоручив named займатися пошуком потрібного сервера доменних імен [6]. Згідно RFC -1035 [11], resolver і сам може опитувати віддалені сервери доменних імен і отримувати від них

відповіді на свої запити. У багатьох Unix - системах саме так воно і зроблено [3]. В цьому випадку resolver звертається до локального сервера доменних імен, отримує від нього адресу одного з серверів домену root, опитує сервер домену root, отримує від нього адреса віддаленого сервера потрібної зони, опитує цей віддалений сервер і отримує IP-адреса якщо він посилав, так званий "прямий" запит.

Як видно з цієї схеми resolver сам знайшов потрібний IP-адреса. Однак найчастіше resolver не породжує нерекурсивних запитів, а переадресовує їх локального сервера доменних імен. Крім цього і локальний сервер і resolver не всі запити виконують за зазначеною процедурою. Справа в тому, що існує кеш, який використовується для зберігання в ньому отриманої від віддаленого сервера інформації (рис. 2.5).

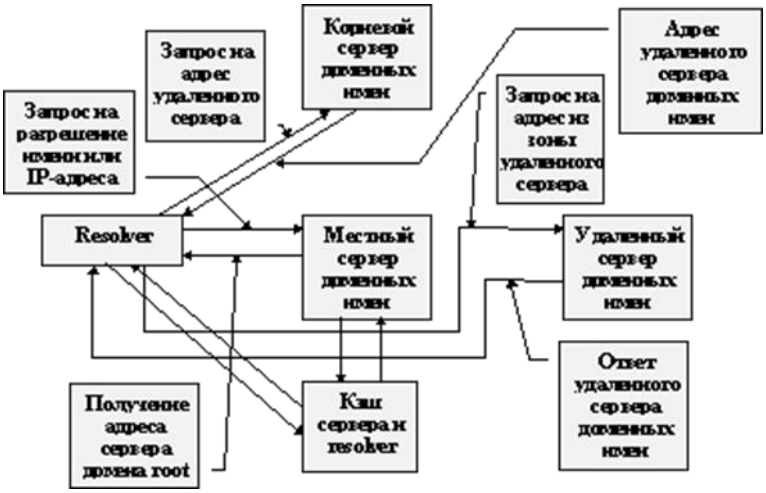


Рисунок 2.5 - Схема дозволу запитів з кешуванням відповідей

Якщо користувач звертається напротязі короткого часу до одного і тому ж ресурсу мережі, то запит на віддалений сервер не відправляється, а інформація шукається в кеші. Взагалі кажучи, прядок обробки запитів можна описати таким чином:

- Пошук відповіді в локальному кеші;
- Пошук відповіді на локальному сервері;
- Пошук інформації в мережі.

При чому кеш може бути як у resolver ‘а, так і у сервера, але найчастіше ці тимчасові сховища інформації про систему DNS об’єднують [8].

Розглянемо тепер нерекурсивний запит прикладної програми до сервера доменних імен на отримання IP-адреси по доменному імені в домені, який знаходиться у віданні віддаленого сервера доменних імен, тобто сервера відмінного від того, домену якого належить комп'ютер, який здійснює запит. При розгляді цього запиту також будемо спиратися на схему з рисунку 2.3.

У загальному вигляді така схема буде виглядати наступним чином:

1. Прикладна програма звертається до місцевого сервера доменних імен за IP-адрес, повідомляючи йому доменне ім'я.

2. Сервер визначає, що адреса не входить в даний домен і звертається за адресою сервера запитуваної домену до кореневого сервера доменних імен.

3. Кореневий сервер доменних імен повідомляє місцевим сервера доменних імен адресу сервера доменних імен необхідного домену.

4. Місцевий сервер доменних імен запитує віддалений сервер на предмет дозволу запиту свого клієнта (прикладної програми).

5. Віддалений сервер повідомляє IP-адреса місцевим сервера.

6. Місцевий сервер повідомляє IP-адреса прикладній програмі.

Серед зазначених вище прикладів, приклад звернення до фінського ftp - архіву якраз і є ілюстрацією такої багато ступінчастою схеми дозволу «прямого» запиту.

Однак, не завжди при вирішенні запитів місцевий сервер звертається до кореневого сервера. Справа в тому, що, як вказує Крег Річмонд, існує різниця між доменом і зоною [8]. Домен - це все безліч машин, які відносяться до одного і того ж доменному імені. Наприклад, всі машини, які в своєму імені мають постфікси sumdu.edu.ua відносяться до домену edu.ua. Однак, сам домен розбивається на піддомени або, як їх ще називають, зони. У кожної зони може бути свій власний сервер доменних імен. Розбиття домена на зони і організація сервера для кожної із зон називається делегування прав управління зоною відповідного серверу доменних імен, або просто делегуванням зони.

Під час налаштування сервера, в його файлах конфігурації можна безпосередньо прописати адреси серверів зон. В цьому випадку, звернення до

кореневого сервера не виробляються, тому що місцевий сервер сам знає адреси віддалених серверів зон. Природно, що все це відноситься до сервера, який делегує права [8].

Існує ще й інший варіант роботи сервера, коли він не запитує кореневий сервер на предмет адреси віддаленого сервера доменних імен. Це відбувається тоді, коли незадовго до цього сервер вже дозволяв задачу отримання IP-адреси по даному доменному імені. Якщо потрібно отримати IP-адресу, то він буде просто витягнутий з буфера сервера, тому що напротязі певного часу, заданого в конфігурації сервера, цю адресу буде зберігатися в cache -сервер. Якщо потрібна адреса з того ж домену, який був зазначений в одному з імен, які вирішувалися до цього, то сервер не звертатиметься до кореневого сервера, тому що адреса віддаленого сервера для цього домена також зберігається в кеші місцевого сервера.

Взагалі кажучи, поняття «зона» і «домен» носять локальний характер і відображають лише той факт, що при настройках і взаємодії між собою сервери доменних імен виходять з дворівневої ієрархії. Це означає, що якщо в зоні необхідно створити поділ на групи, то зону можна назвати доменом, і в ньому організувати нові зони. Наприклад, у компанії Xiaomi LTD є зона en в домені miui.com (en.miui.com). Так ось, цю зону теж можна розбити на зони, наприклад, info.en.miui.com і market.en.miui.com.

Крім нерекурсивною процедури розпізнавання імен можлива ще й рекурсивна процедура розпізнавання імен. Її відмінність від описаної вище нерекурсивною процедури полягає в тому, що віддалений сервер сам опитує свої сервери зон, а не доводить їх до відома адреси місцевим сервера доменних імен. Розглянемо ці два випадки більш докладно.

На рисунку 2.6 продемонстрована нерекурсивна процедура дозволу IP-адреси по доменному імені. Основне навантаження в цьому випадку лягає на місцевий сервер доменних імен, який здійснює опитування всіх інших серверів. Для того, щоб скоротити число таких обмінів, якщо дозволяє обсяг

оперативної пам'яті, можна дозволити буферизацию (кешування) адрес. У цьому випадку число обмінів з віддаленими серверами скоротиться.

На рисунку 2.7 віддалений сервер домену сам вирішує запит на отримання IP-адреси хоста свого домену, використовуючи при цьому нерекурсивний опитування своїх серверів піддоменів.

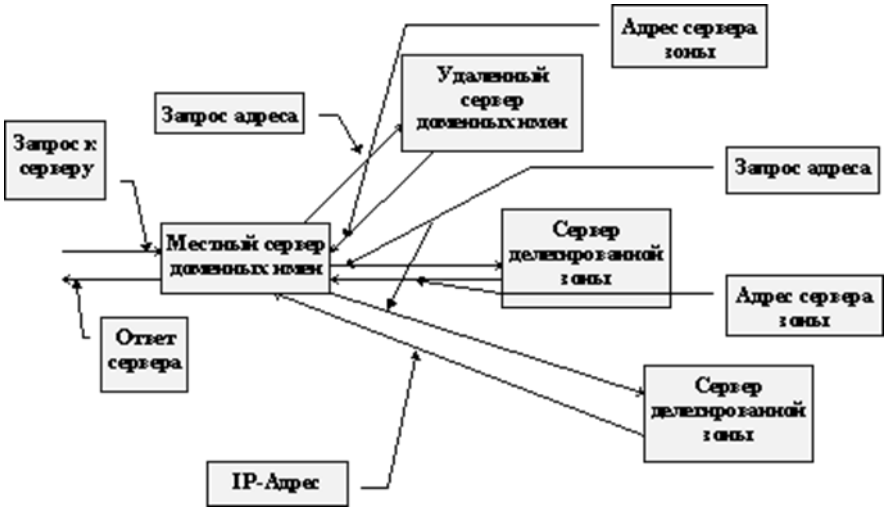


Рисунок 2.6 - Нерекурсивна обробка запиту на отримання IP-адреси

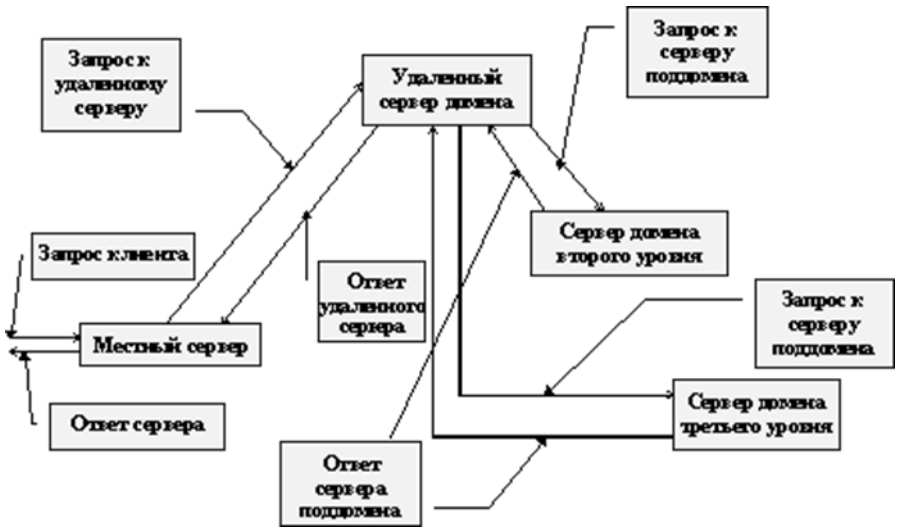


Рисунок 2.7 - Рекурсивна і нерекурсивна процедури дозволу адреси по IP

При цьому локальний сервер відразу отримує від віддаленого сервера адреса хоста, а не адреси серверів піддоменів. Останній спосіб отримання адреси називають рекурсивним, тобто локальний і віддалений сервери взаємодіють з рекурсивної схемою, а віддалений і сервери піддоменів по нерекурсивною.

Тепер від загальних міркувань перейдемо до опису налаштувань системи розв'язання доменних імен в BIND [8].

Налаштування resolver

Як вже говорилося вище, система розв'язання доменних імен IP-адреса і зворотна процедура побудовані за схемою «клієнт-сервер». Функції з бібліотеки libc.a виступають в якості клієнтів, і вся їх сукупність зветься resolver. Для того, щоб клієнт міг звернутися до сервера, він повинен знати наступне:

- Чи встановлений взагалі сервер доменних імен;
- Якщо сервер встановлено, то де (IP-адреса);
- Якщо сервер встановлено, то до якого домену належить машина.

Вся сукупність цих параметрів задається у файлі resolv.conf. Наведемо приклад цього файлу і розберемо призначення кожної з команд, яка може бути використана в файлі настройки resolver.

Зміст файлу resolv.conf.

```
# nonameserver
domain sumdu.edu.ua
nameserver 176.108.235.100
nameserver 212.1.122.90
```

Рядки, що починаються з символу «#» - це коментарі. Серед них слід виділити рядок “nonameserver”. У нашому прикладі він не впливає на роботу системи розв'язання доменних імен, але якщо в мережі немає сервера доменних імен, то слід з цього рядка зняти символ коментарю, а інші команди навпаки перетворити в коментарі. При відсутності сервера доменних імен система буде використовувати файл hosts.

За командою domain система визначає до якого домену вона відноситься. Зазвичай, ім'ям, яке зазначено після команди розширюються неповні імена хостів. Наприклад, якщо звернутися до будь-якої машині тільки по її імені:

telnet fem

то система розширить ім'я fem ім'ям домену та буде шукати машину з ім'ям fem.sumdu.edu.ua.

Зазначений вище приклад - це тільки окремий випадок процедури розширення неповного імені, яка використовується функціями resolver. У загальному вигляді вона виглядає наступним чином:

1. Якщо в прикладній програмі вказано ім'я хоста з точкою на кінці, то розширення імені не проводиться:

ping fem.

В даному випадку ім'я “fem.” завершено точкою.

2. Якщо ім'я вказано без точки, розширення проводиться за такими правилами: або відбувається додавання імені домену, або відбувається звернення до файлу синонімів. Взагалі кажучи, в різних системах це розширення може бути організовано різними способами. Наприклад, HP-UX ім'я фала синонімів вказується у змінній оточення HOSTALIASES. Приклад розширення доменним ім'ям був наведений вище.

3. Якщо в якості імені вказується складене ім'я, то проводиться серія підстановок, за допомогою яких намагаються отримати IP-адресу хоста:

ping fem.sumdu

Якщо в якості домену в resolv.conf вказано домен sumdu.edu.ua, то буде перевірена наступна послідовність імен: fem.sumdu.sumdu.edu.ua; fem.fem.sumdu.edu.ua; fem.sumdu.edu.ua. Останнє ім'я буде дозволено сервером доменних імен.

У назві домену слід також враховувати і те, як буде в цьому випадку працювати вага комплекс програмного забезпечення, встановлений на даному комп'ютері. Справа в тому, що деякі програми, sendmail, наприклад, здатні самі вирішувати проблему визначення IP-адрес по іменах. У ряді випадків це може привести до протиріч і помилок при функціонуванні такого сорту програм.

Команда nameserver визначає адресу сервера доменних імен для домену, в якому дана машина складається. Можливо вказівку кількох серверів. Зазвичай - це не більше трьох серверів доменних імен. У таких системах як Windows NT, наприклад, адміністратор просто не в змозі вказати більше трьох адрес серверів доменних імен.

Порядок у вказівці серверів у файлі resolv.conf визначає порядок опитування серверів. Таким чином, першим в нашому випадку опитуватиметься сервер з адресою 176.108.235.100, а потім сервер з адресою 212.1.122.90. Найбільш доцільно першим вказувати основний сервер доменних імен даного домену, а другим дублюючий сервер доменних імен даного домену.

Як сервер доменних імен можна також вказати і сервер вищого домена, що дозволить підстрахуватися на випадок відмови основного і допоміжного серверів. При цьому не будуть вирішені проблеми адресації машин даного домену, але зате не пропаде можливість звертатися по імені до машин, розташованим за межами домену.

В налаштуваннях resolver треба також враховувати можливість запуску програми named, яка буде працювати як кеш-сервер, що на машинах з «інтелектуальним» resolver 'ом, що дозволяє забезпечити більш ефективну процедуру дозволу запитів до системи DNS.

Програма named.

На жаль досить докладної єдиної документації по named, в якій було б приведено досить велика безліч прикладів опису файлів налаштувань named для різних конфігурацій доменів, в природі (в архівах мережі Internet, та й в книгах, присвячених проблемі адміністрування мереж TCP/IP) не існує. Даний опис також не претендує на повний огляд можливостей named, але дозволяє деяким чином упорядкувати уявлення про ці можливості. Крім того, в даному розділі наводяться приклади з практики настройки цього сервера доменних імен, які можуть бути корисні в практичній роботі і відображають найбільш типові випадки організації доменів [8].

Програма named є одним з найбільш популярних серверів доменних імен з тих, що використовуються в Internet. Для своєї роботи вона використовує 53 порти TCP і UDP.

При розробці named була реалізована концепція функціонування трьох типів серверів доменних імен: primary, secondary, cache. Якщо бути більш точним, то named може одночасно виконувати функції всіх трьох типів серверів.

Primary server - це основний сервер зони. У його базі даних описується відповідність доменних імен і IP-адрес для машин, що належать даній зоні. База даних сервера зберігається на тому ж комп'ютері, де і функціонує сервер доменних імен. При запуску сервера останній звертається до файлів своєї бази даних, після чого він стає здатним відповідати на запити дозволу доменних імен IP-адреса і назад.

Secondary server - це дублюючий сервер зони. Він також здатний відповідати на запити прикладних програм і інших серверів, які вимагають розв'язання доменних імен IP-адреса і/або навпаки, як і primary server. Головна відмінність від primary server полягає в тому, що secondary server не має своєї бази даних зони, а копіює її з primary server в момент свого запуску і потім піклується про підтримку відповідності між скопійованої базою даних і базою даних primary server відповідно до настройками останнього.

Cache server - це сервер, який запам'ятовує в своєму буфері (cache) відповідності між іменами і IP-адреси і зберігає їх деякий час. Якщо користувач достатньо часто звертається до будь-яких ресурсів мережі, то в разі використання cache -сервера він завжди буде швидко отримувати IP-адресу або доменне ім'я, тому що його прикладне програмне забезпечення буде користуватися послугами cache -сервера. При цьому звернення до місцевого сервера зони, кореневого сервера і віддалених серверів зон будуть здійснюватися тільки один раз - в момент першого звернення до ресурсу.

Для організації зони (домену) необхідно мати primary server і, як мінімум один secondary server. Якщо з primary server все більш-менш зрозуміло - він

створюється на комп'ютері, який входить в описуваний домен і управляється адміністратором домену, то розміщення secondary server завжди викликає певні труднощі. Як вже говорилося вище, його можна створити на іншому комп'ютері домену і тим самим формально виконати умови по організації двох серверів доменних імен для однієї зони. Але в цьому рішенні є два нюанси. По-перше, організація другого сервера доменних імен змушує встановлювати або ще одну Unix-машина, або Windows NT, в яких є підтримка BIND. По-друге, з точки зору надійності secondary server найкраще розміщувати у провайдера, тому що зазвичай саме він делегує зону зі свого домену, і через його сервер доменних імен по рекурсивної або нерекурсивною процедурі дозволу імені увесь інший світ отримує доступ до машин вашого домену.

Якщо ж домен розподілений по декількох мереж або подсетям, які до того ж і територіально розташовані в різних будівлях або різних частинах міста, або навіть в різних містах, то тоді для кожної з цих частин домену слід організувати secondary server, що не скасовує розміщення secondary server у провайдерів, які підтримують таку розподілену систему.

У будь-якому випадку secondary server завжди не один, але кожен адміністратор повинен сам вирішувати, скільки їх треба заводити. Крім цього якщо сервер заводиться у комерційного провайдера, то за нього доведеться платити.

Про другий Unix-машина або Windows NT було згадано з тих міркувань, що багато маленькі локальні мережі будуються за схемою, в якій багатозадачна і розрахована на багато користувачів операційна система ставиться тільки на машині-шлюзі, через яку локальна мережа підключається до мережі провайдера. На всіх інших машинах встановлюється або Windows 95, або MS-DOS, одним словом персональна операційна система.

Цілком очевидно, що виділяти ще одну машину для суспільних потреб не дуже хочеться, а свою власну зону мати бажано. Тому, або звертаються до провайдера з запитом на розміщення secondary server, або домовляються з кимось ще.

Будь-який сервер є кешуючий. Primary, і secondary сервери запам'ятовують на деякий час, в своєму буфері (cache) відповідність між іменами і IP-адреси будь-якої зони, якщо до неї хоч раз зверталися, і була виконана процедура дозволу адреси. Це дозволяє скоротити обмін запитами між серверами і забезпечити більш швидке вирішення адреси для часто використовуваних ресурсів.

Організація на базі named тільки кешуючого засноване на тому припущенні, що primary і secondary сервери зони можуть бути перевантажені запитами (типовий випадок для великої організації). У цих умовах час на дозвіл імені може бути досить значним. Кешуючий сервер на вашій обчислювальної установці дозволить дещо скоротити цей час, якщо ви постійно звертаєтеся до одних і тих же хостів. Чекати доведеться тільки при першому зверненні, коли буде виконуватися стандартна процедура дозволу адреси, всі наступні звернення будуть задовольнятися вашим кешуючого.

Якщо число зон, до яких ви звертаєтеся не надто велике, то для них на вашій обчислювальної установці має сенс зробити secondary server для цих зон (якщо звичайно дозволяють ресурси вашого комп'ютера). При цьому ні у кого реєструвати ваш secondary server не треба. В цьому випадку для зазначених зон ви просто створюєте довгостроковий кеш, який обслуговує тільки вашу зону.

Файли налаштування named

Всього існує три типи файлів настройки програми named, або, як їх ще називають, бази даних домену. До них відносяться:

- файл початкового завантаження буфера (cache);
- опису бази даних, він називається named. boot ;
- файли опису «прямий» і «зворотного» зон.

В літературі по named прийнято починати з файлу опису бази даних named - named.boot [8,12].

Цей файл named використовує для своєї настройки і первинного завантаження бази даних домену. Це перше, що шукає named при своєму

запуску. Якщо уважно вивчити скрипти початкового завантаження будь-якого Unix, то при запуску мережевих серверів в частині, що відповідає за запуск сервера доменних імен, можна побачити, що в разі відсутності файлу named.boot програма named НЕ буде запущена.

У файлі named.boot для опису налаштувань named використовуються наступні команди:

· **Directory** - адреса директорії в файловій системі комп'ютера, на якому запускається named.

· **Primary** - визначає зону, для якої даний сервер є primary server, і ім'я файлу бази даних цієї зони. Файл розміщується в директорії, яка вказана в команді directory.

Secondary - визначає зону, для якої даний сервер є secondary server, а також визначає адреси інших серверів для цієї зони, зазвичай primary server, і ім'я файлу де буде вестися копія бази даних цієї зони. Файл розміщується в директорії, зазначеної в команді directory.

Cache - визначає ім'я кеш-файлу. Зазвичай в кеш-файлі описані адреси та імена кореневих серверів, які даний сервер буде використовувати для отримання адрес віддалених серверів доменних імен.

Forwarders - дана команда визначає адреси серверів доменних імен куди слід надсилати не вирішені запити.

Slave - змушує сервер відправляти всі запити на сервери, які визначені командою forwarders.

Перш, ніж розглядати приклади файлів named.boot для різних типів серверів, хотілося б звернути увагу читача на дві останні команди.

Фактично, саме вони і визначають яка з процедур дозволу запиту (рекурсивна або нерекурсивна) будуть застосовуватися на вашому сервері. Якщо у файлі named.boot є команда slave, то визначена рекурсивна процедура дозволу запиту, тому що в цьому випадку named буде відсилати всі запити на сервери зазначені в команді forwarders і від них отримувати адреси або імена віддалених хостів. В цьому випадку сервери з forwarders будуть опитувати

кореневі і віддалені сервери доменів. Якщо ці сервери також містять команду slave, то пошук адреси або імені будуть здійснювати сервери, які визначені в їх файлах named.boot як forwarders.

Розглянемо переваги так налаштованого сервера. Зазвичай так налаштовують сервери піддоменів всередині однієї організації. Як сервер, на який здійснюється пересилання всіх запитів, використовується сервер всього домену. При цьому з кореневими серверами спілкується тільки цей сервер, сервери зон самостійно на кореневі сервери запитів не відправляти. Якщо число машин за межами даного домену, з якими спілкуються користувачі цього сервісу, що не дуже велике, і всі вони компактно розташовані в інших доменах, то центральний сервер домену швидко заповнить свій кеш необхідної для вирішення запитів інформацією. При цьому такий кеш буде тільки один, а працювати він буде з тією ж швидкістю, як якби кожен із серверів зон створював його у себе [8].

З точки зору самого домену в будь-якому випадку сервер зони звертається до сервера домена за адресами з іншої зони, якщо тільки сервер іншої зони не вказано в файлах бази даних named.

Наведемо тепер приклад файлу named.boot, в якому проілюструємо використання зазначених вище команд.

Зміст файлу named.boot

“; An Example of the named.boot;

directory namedb

primary 0. 0.127.in -addr.arpa localhost

primary sumdu.edu.ua sumdu

primary 100. 235.108.176.in -addr.arpa sumdu

secondary ns2.sumdu.edu.ua 212.1.122.90 ns2

cache.named.root”

Зазвичай, файл named.boot розміщується в директорії /etc. Від цієї директорії потім йде відлік місця компонентів бази даних named. У нашому випадку цю базу даних можна представити у вигляді графа (рисунок 2.8), у

якого в якості кореня виступає директорія /etc. В /etc існує піддиректорія namedb, в якій розташовуються всі інші файли.

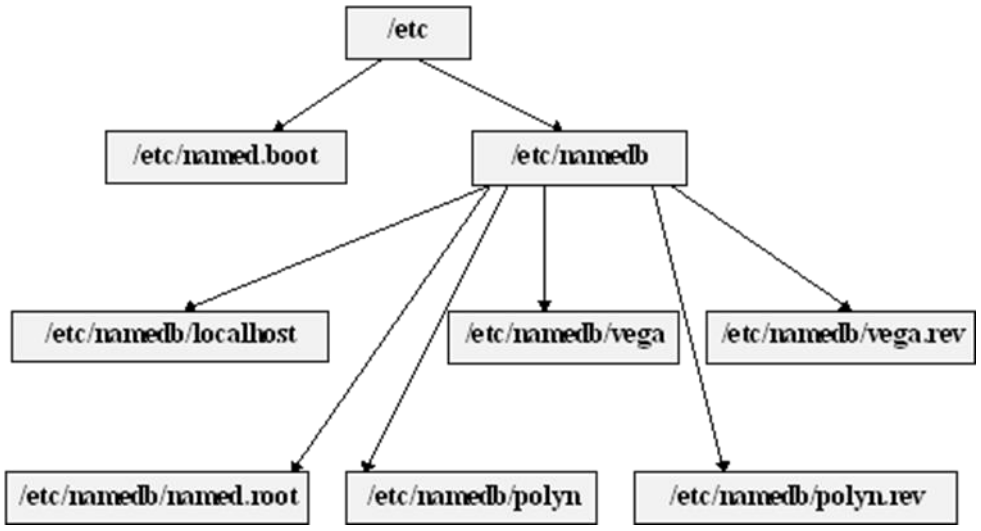


Рисунок 2.8 - Структура розміщення файлів бази даних named

Зіставивши рис. 2.8 і опис з прикладу, легко здогадатися, що остання колонка в кожній з команд опису налаштувань named закінчується ім'ям відповідного файлу або директорії. Розглянемо формат кожної команди більш докладно.

Команда directory визначає директорію namedb як директорію розміщення файлів бази даних named. Взагалі кажучи, зовсім не так вже обов'язково розміщувати всі файли в одній директорії. Можна створити кілька директорій, наприклад, за кількістю зон. Christophe Wolfhugel [10], наприклад, для кожної зони використовує окрему піддиректорію в кореневій директорії named. Якщо дотримуватися цієї логіки розміщення файлів, то отримаємо наступне.

Зміст файлу named.boot при розміщенні файлів опису зон для primary і secondary серверів за різними теками

«; An Example of the named.boot

directory namedb

primary 0. 0.127.in -addr.arpa localhost

primary sumdu.edu.ua s/sumdu

primary 100. 235.108.176.in -addr.arpa sumdu s/sumdu

secondary ns2.sumdu.edu.ua 212.1.122.90 ns2 n/ns2

cache.named. root»

В директорії namedb визначені дві піддиректорії **s** і **n**. В директорії **s** розміщуються файли зон sumdu.edu.ua і 100. 235.108.176.in-addr.arpa, для яких даний сервер є primary. У свою чергу в директорії **n** розміщуються файли опису зон ns2.sumdu.edu.ua і 122.1.212.in-addr.arpa, для яких даний сервер є secondary сервером. Якщо уявити структуру файлів у вигляді графа, то вийде граф, як показано на рисунку 2.9.

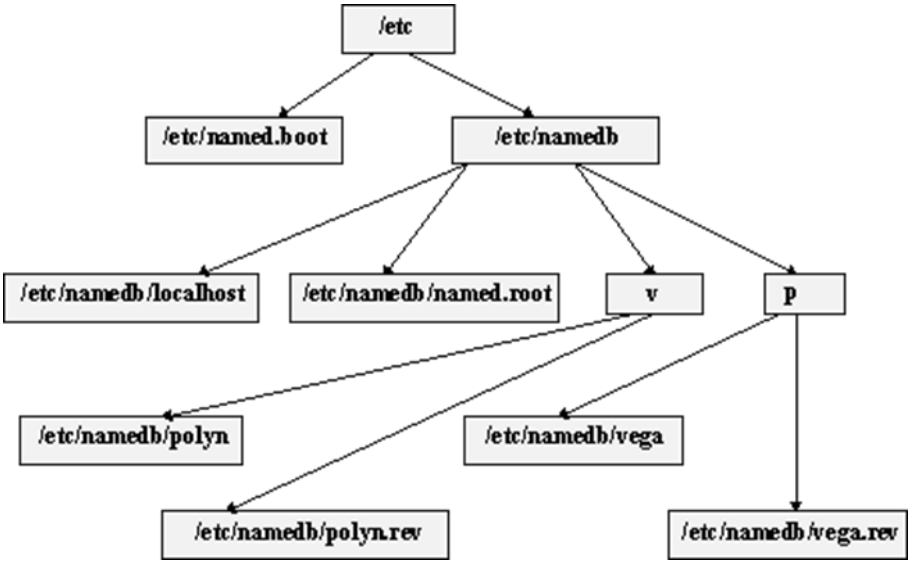


Рисунок 2.9 - Дерево файлів опису бази даних named

Взагалі кажучи, коренем опису файлів бази даних named може бути директорія відмінна від /etc. Якщо програму named запустити з прапором **«-f»,** то в якості параметра можна вказати повний шлях до файлу named.boot або до іншого файлу, який використовується замість named.boot.

У цьому випадку в якості кореневої директорії для файлів опису бази даних named буде використовуватися директорія /usr/paul, а в якості файлу named.boot файл named. par з цієї директорії.

Крім того, в команді directory, як це визначено в наших прикладах, використовується, так званий, неповний шлях до директорії, де розташовані файли бази даних named. Однак, можна вказувати і повний шлях, що дає можливість розташувати ці файли де завгодно в файлової системі сервера.

Наприклад, якщо файли розташовані в директорії /usr/local/etc/namedb, то у файлі named.boot слід вказати наступну команду directory:

«directory/usr/local/etc/namedb»

Команда primary використовується для вказівки зони, для якої даний сервер виступає як primary сервер, і вказівки імені файлу, який цю зону описує. Формат команди primary можна описати таким чином [8]:

«primary <ім'я зони> <ім'я файлу опису зони>»

У прикладах використовується три команди primary. Перша описує «зворотний» зону 0.0.127.in-addr.arpa, друга команда описує «пряму» зону sumdu.edu.ua і третя команда описує «зворотний» зону 43.226.194.in-addr.arpa. Наявність двох «зворотних» зон викликано тим, що пряма зона визначена на двох мережах – 176.108.235.0 і 127.0.0.0.

Мережа 127.0.0.0 - це особлива мережу, тому перша команда повинна бути на будь-якому сервері доменних імен, тому що вона служить для визначення зворотної зони 0.0.127.in-addr.arpa, яка закріплена за будь-якою машиною, на якій встановлений стек протоколів TCP/IP. Особливе призначення цього домену випливає з особливого значення IP-адрес, які закріплені за ним. Вони позначають «петлю», тобто при відправці пакетів за адресою, наприклад, 127.0.0.1 пакети не виходять за межі одного комп'ютера, тобто вони не потрапляють в реальну мережу. У деяких реалізаціях стека певне значення мають і інші адреси мережі 127, наприклад, 127.0.0.2 і 127.0.0.3 в HP-UX.

Дуже часто в прикладах опису файлу named.boot можна зустріти рядок:

«primary 0. 0.127.in-addr.arpa 0.0.127.in-addr.arpa»

Повторення імені домена в якості другого аргументу означає тільки те, що в директорії файлів опису бази даних named повинен бути файл з таким ім'ям. Для тих, хто звик до того, що в файлової системі MS-DOS дозволені тільки трьохбуквені розширення імені файлу, подібне ім'я виглядає досить дивно, але у ентузіастів Unix це не повинно викликати труднощів.

Використання імен зон в якості імен файлів - це загальноприйнята практика. Так набагато простіше орієнтуватися серед файлів опису бази даних named.

Часто замість 0.0.127.in-addr.arpa вказують 127.in-addr.arpa. Мережа 127 - це мережа класу А. Для цієї мережі що 127, що 127.0, що 127.0.0 - суть одне і теж. Так як при вказівці зворотного зони числа в IP-адресах вказуються в зворотній послідовності, то 0.0.127.in-addr.arpa і 127.in-addr.arpa також означають одне і теж. Взагалі кажучи, обробка цієї зворотної зони згідно RFC-1035 [11] проводиться особливим способом.

Серед фахівців з named немає єдності думок про стилі визначення прямих і зворотних зон, в тому числі, і про зону 0.0.127.in-addr.arpa. Деякі пропонують ввести «пряму» зону, яка б відповідала «зворотної» зоні 0.0.127.in-addr.arpa. Пов'язано це з доменним ім'ям, яке ставиться у відповідність з адресою 127.0.0.1. Але до цього питання повернемося при обговоренні змісту самих файлів опису зон.

Команда secondary використовується для опису зон, для яких даний сервер є secondary сервером, і має такий вигляд [8]:

«secondary <ім'я зони> <список IP-адрес серверів зони> <ім'я файлу опису зони>»

У наших прикладах описано дві зони, для яких даний сервер є secondary сервером – ns2.sumdu.edu.ua і 122.1.212.in-addr.arpa. У прикладах для кожної з цих зон вказано за два IP-адреси серверів, які описують зону. Взагалі кажучи, досить вказувати адресу тільки primary сервера для зони. З точки зору актуальності стану бази даних зони, для якої створюється secondary сервер, вказівка одного тільки primary найбільш правильне рішення, тому що тільки primary сервер містить найбільш актуальну базу даних зони. Однак, в ряді випадків, має сенс вказати кілька серверів, primary і secondary, наприклад, у разі зазначення кількох серверів, база копіюється з того, що зазначений першим, якщо він доступний. Якщо сервер не доступний, то опитується наступний у списку, до першого доступного сервера.

Файл, який вказаний останнім аргументом в команді secondary, створюється named при запуску і постійно оновлюється відповідно до опису взаємодії primary і secondary серверів. Редагувати вручну цей файл не має сенсу, тому що це калька з primary сервера, і через постійні проміжки часу цей файл оновлюється програмою named. Таким чином, якщо хто-небудь і внесе до нього зміни, то named, створюючи нову копію бази даних зони, затре ці зміни.

На відміну від primary, secondary сервер не здатний підтримувати дозвіл запитів як завгодно довго. Як тільки закінчиться час актуальності даних в його базі даних, він намагається створити нову копію бази з бази даних primary сервера. Якщо це не вдається, то сервер перестає обслуговувати зону.

Головне призначення secondary сервера - це підвищення надійності служби доменних імен. Опис зони named копіює з серверів, зазначених в якості аргументу команди secondary. Там вказані не тільки primary сервер, але і інші secondary сервери. Зона копіюється з того сервера, який доступний. Це означає, що на даному сервері може виявитися копія з іншого secondary сервера, що, взагалі кажучи, не дуже добре, якщо мова йде про сервер, який дійсно призначається для дублювання зони.

На самому початку опису BIND було сказано, що сервіс працює по 53 портів UDP і TCP. Справа в тому, що звичайні запити на дозвіл імен IP-адреса або IP-адрес іменами відправляються по транспорту UDP. Це робиться через те, що обсяг інформації, що передається невеликий. Але при організації secondary сервера ситуація змінюється. Цей сервер запитує інший сервер, прописаний в команді secondary, цілком все опис зони, а це може бути досить великий обсяг інформації. Ось в цьому випадку і використовується сервіс TCP. Через таку архітектури виникають багато проблем пов'язані як зі швидкістю обслуговування запитів до системи доменних імен, так і з безпекою мережі взагалі [1].

При встановленні великих тимчасових інтервалів очікування відповідей відмову на обслуговування приходить тільки після закінчення часу очікування

для даного сервісу UDP. Це якщо сервіс неактивний. Якщо сервіс активний, то не можна точно встановити працює він чи ні, тому що сервіс UDP - це сервіс без встановлення з'єднання. Але якщо говорити про копіювання зон, то можна зіткнутися з такою ситуацією: запити дозволяються, а зони не копіюються. У цьому випадку крім адміністратора даного сервера ніхто нічого сказати не може. Адміністратор може навмисно заборонити копіювання зони, а може бути за відведений інтервал часу не вдається передати зону, або під час передачі розривається TCP сесія. Для української частини Internet таке трапляється досить часто. Якщо раптом користувачам здається, що система починає повільно «вмирати», то це може відбуватися через те, що secondary сервера не можуть оновити опису зон, а так як копії можуть бути отримані в різний час, то сервери перестають працювати в різний час.

Команда cache служить для визначення файлу з початковими даними для запуску named. Для того, щоб почати відповідати на запити named повинна знати адреси інших серверів доменних імен, до яких можна було б звернутися із запитами на дозвіл IP-адреси по імені або імені по IP-адреси.

Формат команди виглядає наступним чином [8]:

«cache <ім'я зони> <ім'я файлу cache>»

Зазвичай обговорення cache зводиться до обговорення того, які кореневі сервери повинні бути вказані в файлі cache, і як підтримувати актуальність цього файлу. Перш, ніж звернутися до формату команди, зауважу, що не тільки кореневі сервери можуть зазначатися в файлі cache, але також і інші сервери, які часто використовуються для вирішення запитів в вашому домені.

Згідно з матеріалом Крега Річмонда (Craig Richmond) [8] різні версії named по різному використовують файл, вказаний в команді named. Одні програми, завантаживши дані з цього файлу, більше його не використовують, інші навпаки, постійно вносять до нього зміни.

У разі стабільного файлу адміністратор системи повинен сам піклуватися про його актуальності. Для цього він повинен регулярно перевіряти відповідність між його файлом і файлом, який підтримується в

ns.internic.net. Отримати копію файлу можна або, з ftp -сервера *ftp.rs.internic.net*, або по команді:

«dig @ ns.internic.net.ns > root.cache»

Том Ягер (Tom Yager) [1] рекомендує інше джерело отримання cache - ftp://nic.ddn.mil/netinfo/root-servers.txt.

Файл cache (лютий 1996 року):

«; Servers from the root domain

; ftp://nic.ddn.mil/netinfo/root-servers.txt

. 99999999 IN NS A.ROOT-SERVERS.NET

. 99999999 IN NS B.ROOT-SERVERS.NET

. 99999999 IN NS C.ROOT-SERVERS.NET

. 99999999 IN NS D.ROOT-SERVERS.NET

. 99999999 IN NS E.ROOT-SERVERS.NET

. 99999999 IN NS F.ROOT-SERVERS.NET

. 99999999 IN NS G.ROOT-SERVERS.NET

. 99999999 IN NS H.ROOT-SERVERS.NET

. 99999999 IN NS I.ROOT-SERVERS.NET

; Root servers by address

A.ROOT-SERVERS.NET 99999999 IN A 198.41.0.4

B.ROOT-SERVERS.NET 99999999 IN A 128.9.0.107

C.ROOT-SERVERS.NET 99999999 IN A 192.33.4.12

D.ROOT-SERVERS.NET 99999999 IN A 128.8.10.90

E.ROOT-SERVERS.NET 99999999 IN A 192.203.230.10

F.ROOT-SERVERS.NET 99999999 IN A 192.5.5.241

G.ROOT-SERVERS.NET 99999999 IN A 192.112.36.4

H.ROOT-SERVERS.NET 99999999 IN A 128.63.2.53

I.ROOT-SERVERS.NET 99999999 IN A 192.36.148.17»

Все, що було сказано про зміст файлу з команди cache відноситься до випадку, коли в якості імені зони вказана «.». Це означає, що описується коренева зона, що в свою чергу означає опис кореневих серверів доменних

імен, до яких named звертається при роботі в режимі нерекурсивною процедури дозволу запитів на IP-адреса по імені або запитів на ім'я по IP-адреси.

Якщо в файл chache необхідно внести іншу інформацію, то це робиться аналогічно опису кореневих серверів. Реалізація файлу cache, відмінного від зазначеного в прикладі буде розглянута для випадку делегування зони в домені.

Команда **forwarders** визначає IP-адреси серверів, на які слід відправляти недозволені даними сервером запити. Команда має наступний вигляд [8]:

«forwarders <список IP-адрес серверів>»

Випадок організації рекурсивної процедури дозволу імені з використанням цієї команди був розглянутий раніше. Однак цим випадком не обмежується коло використання команди forwarders. При реєстрації домену деякий час зовнішній (щодо цього домену) світ не підозрює про існування домену. Повинен пройти якийсь час, перш ніж буде закінчена процедура реєстрації домена і оновлення баз даних вищого в ієрархії DNS домену на всіх серверах як primary, так і secondary. Однак всередині домену все працює нормально, тому що сервер запускається до офіційної реєстрації і здатний обслуговувати машини домену. Однак, він може і не знати інформації про всіх доменах Internet. З цієї причини завжди корисно вказати команду forwarders на сервер домену вищого щодо даного домену. Так, наприклад, для зони sumdu.edu.ua сервером домена edu.uaa, в який входить дана зона, є сервер з IP-адреси 8.8.8.8. Для того, щоб пересилати на нього запити на дозвіл імен IP-адреси в named.boot слід включити команду:

«forwarders 8.8.8.8»

Зазвичай вказують не один, а кілька IP-адрес серверів, які в змозі відповісти на запити клієнтів, на які даний сервер відповісти не може. Наприклад, для сервера зони sumdu.edu.ua, який запущений, але ще не зареєстрований, можна вказати два сервера:

«forwarders 8.8.8.8 176.108.235.100»

Команда slave вказується тоді, коли сервер спілкується із зовнішнім світом через сервери-посередники, зазначені в команді forwarders. Параметрів у даній команди немає. Файл named.boot для того сервера, якщо він ще й primary сервер для зон sumdu.edu.ua і 43.226.194.in-addr. arpa буде виглядати так, як показано в прикладі.

Підлеглий сервер, що працює по рекурсивної процедури дозволу запитів від resolver

«; An Example of the named.boot

directory namedb

primary 0. 0.127.in -addr.arpa localhost

primary sumdu.edu.ua sumdu

primary 235. 108.176.in -addr.arpa sumdu

cache. named. root

forwarders 8.8.8.8 176.108.235.100 slave»

Фактично, команда slave дозволяє організувати, в деякому сенсі, «інтелектуальний» resolver.

Перейдемо тепер до опису змісту файлів бази даних named. Всі ці файли складаються із записів, які мають однаковий формат і називаються записами опису ресурсів.

Формат запису визначається документом RFC -1033 [10], і має такий вигляд:

«<name> < ttl > <class> <type> <data>» [10]

Всі поля відокремлюються одна від одної пропуском або табуляцією. Кожне з них має таке значення:

Поле name - це ім'я об'єкта. Об'єктом може бути хост, домен і піддомен. Існують спеціальні правила іменування об'єктів, які базуються на понятті поточного імені домену. Програма named аналізує файл бази даних, починаючи з першого запису послідовно до останнього запису файлу. Спочатку поточним ім'ям домену є ім'я, вказане в командах primary, secondary

або cache файлу named.boot, в залежності від того про який файлі бази даних named йдеться, якщо перший запис цього файлу містить ім'я @. В іншому випадку для визначення поточного імені повинна бути вказана команда $ORIGIN**.** Якщо ім'я в запису опису ресурсу опущено, то ресурс відноситься до поточного імені домену. Якщо ім'я вказано без точки на кінці, то воно розширюється поточним ім'ям домену. Для зміни поточного імені домену слід ввести або команду $ORIGIN, або вказати ім'я записи ресурсу з точкою на кінці. Якщо в якості імені вказана одна крапка («.») або дві крапки («..») то такий запис описує домен root, тобто кореневої домен. Якщо в імені записи зустрічається символ «*», то це він означає що замість нього можна мати на увазі будь-яку дозволену послідовність символів. В англомовних джерелах це називають «wildcard character». Для користувачів будь-якій операційній системи вживання цього символу добре знайоме по командам dir (MS-DOS) або ls (Unix [3]). Наприклад, при необхідності отримати список файлів, що закінчуються розширенням «bak», видається команда:

“ls*.bak”

Точно також використовується цей символ і в імені записи бази даних опису домену. Якщо в поле імені вказано тільки символ «*», то це передбачає *«*.<Ім'я поточного домену>».* Якщо за «*» слід ім'я, то воно обмежує розширення імен «*». Наприклад,«*.sumdu.edu.ua.» визначає всі імена з домену polyn.ssu.edu.ua, в тому числі і імена машин в піддоменів, а не тільки імена машин в домені і імена самих піддоменів. Найбільш часто «*» використовується в записах MX, які регулюють обмін електронною поштою.

Поле ttl - це поле визначає час (в секундах), напротязі якого даний запис зберігається в кеші. Значення поля задається у вигляді восьми десяткових цифр, таким чином, максимальне значення ttl - 99999999. Саме це значення і задається для кореневих серверів доменних імен в прикладі. Якщо залишити це поле пустим, то за замовчуванням приймається значення, вказане в параметрі minimum поля даних (data) записи SOA для даної зони.

Поле class визначає клас записи опису ресурсу. В Internet використовується тільки один клас записів - клас IN. В принципі існують ще класи HS (Hessiod) і CH (Chaos) [10,11]. Деякі автори, наприклад, Ronny H. Kavli [3], який написав коментарі до FAQ Kaig Richmond [4], вважає, що наявність цього поля тільки «замутняют чистий лик» бази даних опису ресурсів DNS. У будь-якому випадку всі записи з бази даних named мають вигляд [8]:

“<name> < ttl > IN <type> <data>” [8]

Поле type визначає тип запису опису ресурсів. До таким типам ставляться : SOA (Start Of Authority), NS (Name Server), A (Address), MX (Mail eXchanger) CNAME (Canonical NAME), WKS (Well Known Services), PTR (PoinTeR), HINFO (Host INFOrmation). Перераховані вище типи записів складають набір стандартних записів, які можуть зустрітися в базі даних опису домену. Крім цих типів існують ще й експериментальні типи. Всі вони стосуються, головним чином, можливостей управління поштою. Слід зауважити, що не всі стандартні записи використовуються в базах даних опису доменів. Так російські адміністратори рідко користуються записами типу HINFO і WKS. Деякі адміністратори вважають, що замість CNAME краще призначити ще одну адресу через запис типу A, існують і інші переваги.

В поле data вказуються дані для кожного запису опису ресурсів. Для різних типів записів формат даних також різний і відповідає призначенню записи опису ресурсів.

Крім записів опису ресурсів в файлах опису бази даних домену можуть зустрічатися і інші записи. Найпоширеніші з них - це записи коментарів. Запис коментаря починається з символу ";" в першій позиції рядка. Все, що слід до кінця рядка, розглядається в цьому випадку як коментар. Якщо цей символ зустрінеться в середині рядка, то програма передбачає, що всі, що слід далі до кінця рядка - це коментар. При розборі записів опису ресурсів я постараюся звернути увагу на використання коментарів.

Якщо дані не поміщаються в межі екрану, то можна використовувати продовження опису записи на інший рядок. Я не пишу тут «якщо дані не поміщаються на одному рядку», тому що це буде не вірно з точки зору подання файла бази даних в файлової системі. В принципі довжина рядка може мати досить велику довжину, достатню для того, щоб вмістити опис записи ресурсів, але це незручно для редагування файлу бази даних, тому що довжина відображається рядки на екрані більшості алфавітно-цифрових дисплеїв обмежена 80-ма символами. В описі файлу бази даних домену не передбачений символ продовження записи типу «\» в csh Unix [3]. Як механізм продовження записів використовується пара дужок «(», «)». Якщо в деякій рядку зустрілася відкриває дужка «(«, то всі дані до закриває дужки будуть приписані до цієї рядку.

Іншим механізмом, який використовується при описі елементів запису опису ресурсу, є маскування символів. Маскування символів застосовується в тому, випадку, якщо необхідно використовувати символ, який має особливе значення для записів опису ресурсів. Наприклад, символ «@». Для цього використовується символ зворотного слеша «\». Аналогічно мови програмування С символ можна вказувати і цифрами. Тільки в цьому випадку використовується десяткове число, яке відповідає коду ASCII, наприклад, «\ 040» також позначає символ «@».

2.3 Типи записів ДНС

*Запис "*Start Of Authority"

Опис записів треба почати ззапису SOA (Start Of Authority). Запис SOA відзначає початок опису зони. Зазвичай, це перший запис опису зони. Деякі автори, зокрема, згадуваний раніше Kavli [8], рекомендують наявність однієї і тільки однієї записи типу SOA в кожному файлі, який вказаний в запису primary файлу named. boot.

Формат запису SOA можна уявити як [8]:

"[zone] [ttl] IN SOA origin contact (serial refresh retry expire minimum)" [8]

У цьому записі кожне з полів означає наступне:

Поле zone - ім'я зони. Якщо мова йде про зону, описаної в запису primary файлу named. boot, то в якості імені вживається символ комерційного ей - «@». У цьому випадку в якості імені поточної зони береться ім'я, вказане в якості першого аргументу команди primary з файлу named. boot. Поле зони обов'язково повинно бути вказано. Інакше named не зможе прив'язати наступні за даним записом опису до імені зони. Пусте ім'я зони не є допустимим.

Поле ttl в запису SOA завжди порожнє. Справа в тому, що час кешування для записів опису зони задається останнім аргументом даних записи SOA.

Поле origin - це доменне ім'я primary сервера зони. У разі опису зони sumdu.edu.ua в якості сервера використовується машина sumdu – gw.sumdu.edu.ua. Дане доменне ім'я і має бути зазначено в поле origin. Дуже часто в цьому полі можна зустріти імена, які починаються з “ns”, наприклад, ns.gov.ua або ns.google.com. В даному випадку це означає тільки те, в даних зонах існують машини з ім'ям “ns”, і на них розміщені primary сервери цих зон. Ніякого спеціального зарезервованого імені для вказівки в поле origin немає. Використання, зазначених вище імен обгрунтовано тим, що їх просто легше запам'ятати, тому що “NS” означає “name server”.

Поле **contact** визначає поштову адресу особи, яка здійснює адміністрування зони. Ця електронна адреса має збігатися зі значенням адреси зазначеним в заявці на домен в поле “zone - c :». Є, однак, одна особливість при вказівці цієї адреси. Так як символ «@» має особливий сенс при описі зони, то замість цього символу в поштовій адресі використовується символ «.». Наприклад, якщо ваш покірний слуга виступає адміністратором домену, то в поле contact слід писати не *doc@sumdu.edu.ua,* а *doc.sumdu.edu.ua. Я*кщо в імені користувача є будь-які особливі символи, які мають спеціальні значення при описі зони, то вони повинні маскуватися символом зворотного слеша - «\». Типовий приклад - поштову адресу типу *veliar@veliar.pro.* В цьому випадку точку слід замаскувати і написати в поле contact : *veliar\.veliar.veliar.pro.*

Діючи, таким чином, ми виключили особливе значення точки як роздільник піддоменів і забезпечили інтерпретацію імені як єдиної символьного рядка.

Поле даних у записі SOA розбите на аргументи, які визначають порядок роботи сервера з записами опису зони. Як правило, всі аргументи мають у своєму розпорядженні на іншому рядку або, для кращого відображення кожен на своєму рядку, що змушує записувати їх усередині дужок.

Атрибут serial - визначає серійний номер файлу зони. Якщо говорити простіше, то в цьому полі ведеться облік змін файлу опису зони. Поле може складатися з восьми десяткових цифр. В принципі це можуть бути будь-які числа, але найчастіше адміністратори використовують в якості серійного номера рік, місяць і день внесення змін до файл опису зони. Чесно кажучи, я вважаю за краще просто порядкові імена. На даний момент наша мережа досить інтенсивно росте, що змушує робити зміни по кілька разів на день. Просте збільшення номера вирішує всі ці проблеми. Важливість серійного номера визначається тим, що коли вторинний (secondary) сервер звертається до первинного (primary) сервера для оновлення інформації про зону, то він порівнює серійний номер зі свого кеша з серійним номером з бази даних первинного сервера. Якщо серійний номер з primary сервер більше, то secondary сервер звертається до primary і копіює опис всієї зони цілком, якщо немає, то він не вносить змін в свою базу даних. Таким чином, коли виробляються зміни в базі даних primary сервера, то значення атрибута serial в поле даних записи SOA для зони, опис якої було змінено, має бути збільшено. Незміненому номера - це типова помилка, на якій автор не раз себе ловив.

Атрибут refresh визначає інтервал часу, після якого secondary сервер зобов'язаний звернутися до primary сервера з запитом на верифікацію свого опису зони. Як вже раніше було сказано, при цьому перевіряється серійний номер опису зони. Опис зони завантажується secondary сервером кожного разу, коли він стартує або перевантажується. Однак, при стабільній роботі може пройти досить великий інтервал часу, поки ці події не відбудуться в дійсності. Крім того, більшість систем, які підтримують сервіс доменних імен,

працюють цілодобово. Отже, необхідний механізм синхронізації баз даних primary і secondary серверів. Поле refresh задає інтервал, після якого ця синхронізація автоматично виконується. Тривалість інтервалу вказується в секундах. В поле refresh можна вказати до восьми цифр. Вказівка маленької цифри призведе до невиправданої завантаженні мережі, адже в більшості випадків опис зон досить стабільно, але в ряді випадків, коли зона тільки організовується, можна вказати і невеликий інтервал. Найбільш типовим є синхронізація стану опису зон один (86400) - два (43200) рази на добу.

Атрибут retry починає грати роль тоді, коли primary сервер з якої-небудь причини не здатний задовольнити запит secondary сервера за час певне атрибутом refresh. А точніше, в момент настання часу синхронізації опису зони, primary сервер не відповідає на запити secondary сервера. Атрибут retry визначає інтервал часу після якого secondary сервер повинен повторити спробу синхронізувати опис зони з primary сервером. Значення цього атрибута також може складатися максимум з восьми цифр. При установці цього значення до уваги слід приймати кілька факторів. По-перше, якщо primary сервер не відповідає, то, швидше за все сталося щось серйозне (обробники вирізали частину мережі, тому що заважала фарбувати стіни, екскаватор перекопав магістраль або відключили живлення в мережі). Така причина не може бути швидко усунена, тому установка занадто малого часу опитування просто даремно навантажує мережу. По-друге, якщо на primary сервер прописано багато зон, і він обслуговує велику кількість запитів, то він може просто не встигнути відповісти на запит, а дуже часті запити від secondary сервера просто «підливають масло у вогонь», погіршуючи і так повільну роботу сервера. Зазвичай значення цього атрибута встановлюють рівним одній годині (3600).

Атрибут expire визначає інтервал часу, після якого secondary повинен припинити обслуговування запитів до зони, якщо він не зміг напротязі цього часу верифікувати опис зони, використовуючи інформацію з primary сервера. Зазвичай це інтервал роблять досить великим, скажімо півтора місяця

(3888000). Якщо зробити маленький інтервал, то який сенс в secondary сервері, якщо він раптово помре після відключення primary сервера. Втечение всього цього часу secondary сервер буде намагатися встановити контакт с primary сервером і обслуговувати запити до зони. Правда всі ці міркування хороші для випадку, коли просто відключиться або ламається машина, на якій стоїть primary сервер. Якщо ж відключиться вся мережа, яка описана в зоні, а для більшості організацій так воно і є, то secondary сервер стає подібним космічному апарату «Піонер», який подорожує у Всесвіті, давно втративши будь-який зв'язок із Землею.

Останній атрибут з поля даних записи SOA - minimum**.** Він визначає час зберігання записи опису ресурсу в кеші віддаленої машини, тобто значення поля ttl за замовчуванням для всіх записів опису зони. Саме з цієї причини поле ttl в запису SOA залишається порожнім. Дане поле впливає на те, як часто віддалені сервери будуть звертатися до сервера опису зони за інформацією. Хорошим тоном вважається установка значення в цьому полі не менше, ніж значення в поле expire. Іноді вказують взагалі максимально можливе значення - 99999999. Маленьке значення буде приводити до непродуктивної завантаженні мережі - адже відповідності межу IP-адреси змінюються дуже рідко.

Зверніть увагу на те, що всі атрибути запису SOA визначають порядок взаємодії primary сервера і secondary серверів. Виняток становить тільки атрибут minimum. В даному випадку мова йде про кешуванні адрес на серверах, а системою resolver. Однак, в більшості випадків ця різниця не проявляється. Справа в тому, що в якості resolver на локальної обчислювальної установці виступають функції зі стандартної бібліотеки, які надсилають запити до локального сервера доменних імен і самі не здійснюють кешування відповідностей між іменами і IP-адрес. В цьому випадку кешування здійснюється локальним сервером доменних імен. У тому випадку, коли на машині запускається свій кешуючий сервер, який не описує жодної зони, а

використовується тільки для обслуговування запитів до сервісу доменних імен, саме він і є тією частиною resolver, яка кеширує відповідності.

Таким чином, коли мова йде про атрибут minimum, то найчастіше його описують у контексті програми named, яка може використовуватися не тільки як primary або secondary сервер, але і як кешуючий сервер. У цьому випадку поле ttl записи опису ресурсу і атрибут minimum задають час зберігання записи опису ресурсу в кеші.

Наведемо приклад записи типу SOA :

“sumdu.edu.ua. 0 IN SOA ns.sumdu.edu.ua. admin.sumdu.edu.ua. 2020031034 3600 900 1209600 1”

“$TTL 604800

@ IN SOA localhost. root.localhost. (

2 ; Serial

604800 ; Refresh

86400 ; Retry

2419200 ; Expire

604800) ; Negative Cache TTL”

В даному прикладі ім'я зони береться з запису primary файлу named. boot, тому в поле імені зони вказано символ «@». Як сервер для зони використовується машина з ім'ям sumdu – gw.sumdu.edu.ua. Поштова адреса відповідального за зону - admin@sumdu.edu.ua. Дужка говорить про те, що опис записи SOA буде продовжено на наступних рядках. Кожна з наступних рядків описує один атрибут з поля даних записи SOA. При цьому зверніть увагу на те, що символ «;» маскує весь залишок рядка для інтерпретації програмою named, тобто коментар починається з символу ";" і закінчується символом переходу на новий рядок. Атрибут serial визначає дану 2-ю версію опису зони, атрибут refresh встановлює періодичність опитування primary сервера secondary сервером як один раз на добу, у разі невдачі secondary сервер починає опитувати primary сервер кожну годину і, якщо відновити взаємодію не вдається, то через 45 днів secondary сервер перестає обслуговувати запити

до цієї зони. Інформація про відповідність доменних імен IP-адреса повинна зберігатися в кеші resolver або віддаленого сервера максимально можливий час.

Запис “Name Server”

Запис NS використовується для визначення сервера, який описує зону. Ця запис дозволяє делегувати піддомени, підтримуючи тим самим ієрархію доменів системи доменних імен Internet. Для того, щоб домен був видний в мережі в зоні опису домену старшого рівня повинна бути вказана запис типу NS для цього поддомена. Формат запису NS можна записати у вигляді [8]:

“[Domain] [ttl] IN NS [server]” [8]

В поле domain вказується ім'я домену, для якого сервер, вказаний останнім аргументом записи NS підтримує опис зони. Значення поля ttl зазвичай залишають порожнім, тим самим, змушуючи named встановлювати значення за замовчуванням (поле minimum із запису SOA для даної зони). У назві сервера слід мати на увазі наступні обставини: якщо в якості сервера вказується сервер з поточної зони, то можна обійтися тільки ім'ям машини, не вказуючи повне доменне ім'я цього комп'ютера, тому що ім'я може бути розширено ім'ям домену, але це швидше виняток, ніж правило. У записах NS вказуються як primary, так і secondary сервери. Останні ж, швидше за все, належать іншим зонам, і для них слід вказувати повне доменне ім'я сервера із зазначенням імені машини і імені зони. Наприклад, для сервера з домену sumdu.edu.ua з ім'ям ns слід писати повністю ns.sumdu.edu.ua. Для того, щоб дотримуватися якогось єдиного стилю і щоб уникнути помилок рекомендується писати завжди повне ім'я сервера.

Наведемо приклад записи описує делегування поддомена:

“$ ORIGIN sumdu.edu.ua.

zone1 IN NS ns.sumdu.edu.ua ns2.sumdu.edu.ua”

В даному прикладі в якості першого рядка використовується команда $ ORIGIN, яка визначає ім'я поточної зони. В даному контексті це означає делегування зони в домені sumdu.edu.ua, повне ім'я якої буде zone

1sumdu.edu.ua. Як ім'я сервера, який керує делегованої зоною, зазначено ім'я ns.sumdu.edu.ua, тобто повне ім'я машини в делеговане домені. В принципі, в даному випадку можна було б написати і по-іншому:

“$ ORIGIN sumdu.edu.ua.

zone1 IN NS ns.zone “

тому ім'я належить поточному домену і може бути розширено його ім'ям.

*Адресна запис “*Address”

Основне призначення адресної записи - встановити відповідність між доменним ім'ям машини і IP-адрес. В принципі для різних імен можна встановити один і той же адресу. Адресна запис - це основа файлу опису «прямий» зони. Запис має такий вигляд [8]:

“[host] [ttl] IN A [address]” [8]

Поле host визначає доменне ім'я машини. Найчастіше це ім'я вказується щодо імені поточної зони, тому на кінці імені не проставляється символ «.». Якщо ж треба описати машину з іншої зони, то слід вказати її повне доменне ім'я і не забути в кінці цього імені символ «.». Наприклад, *“elit.sumdu.edu.ua.”*, вказане в полі host посилається на машину з ім'ям *elit.sumdu.edu.ua.* Поле ttl зазвичай залишають порожнім, а в поле address вказують IP-адреса машини. В цю адресу точку на кінці ставити не треба. При делегуванні доменів і на самому початку опису зони виникає проблема опису серверів, що підтримують бази даних опису зони і піддоменів. Адже реально при маршрутизації пакетів по мережі потрібно знати тільки IP-адреси, а їх-то якраз і не вистачає для звернення до серверів піддоменів і до сервера зони, якщо він розташований в тій же зоні. Для вирішення цієї проблеми використовуються «приклеєні» (буквально “glue”) записи типу «A». При цьому взагалі-то немає різниці, що буде зазначено раніше використання доменного імені або визначення відповідності цього імені IP-адреси.

Наведемо приклад записи типу A і використання «приклеєних» адресних записів. По-перше, розглянемо просту запис типу A :

“$ORIGIN sumdu.edu.ua.

elit IN A 176.108.235.105”

В даному випадку в зоні sumdu.edu.ua визначається доменне ім'я для машини з IP-адреси 176.108.235.105. Для кожної машини зони буде вказана така ж запис.

Тепер розглянемо можливість призначення «приклеєною» записи для сервера зони для звернень на ім'я зони, а не конкретної машини, що буває корисно при налаштуванні електронної пошти.

“$ORIGIN sumdu.edu.ua

@ IN SOA sumdu ns.sumdu.edu.ua (

101 ; serial number

86400 ; refresh within a day

3600 ; retry every hour

3888000 ; expire after 45 days

99999999) ; default ttl is set to maximum value

IN NS gw.sumdu.edu.ua.

IN A 176.108.235.100

elit IN A 176.108.235.105”

В даному прикладі при призначенні сервера зони в запису NS IP-адреса цього сервера ще не визначений, але це і не має значення, головне, щоб він був визначений далі в одній з адресних записів визначення. Крім цього, для обслуговування типу:

“/ Usr/paul > mail admin@sumdu.edu.ua”

«Приклеєна» ще одна адресна запис, яка призначає поточного домену IP-адреса 176.108.235.100. Формально цей запис визначає адресу для, так званих, запитів за неповним імені. В даному випадку для всіх запитів, які використовують ім'я зони, визначений адреса шлюзу, на якому встановлено обслуговування всіх сервісів, наприклад, *http://sumdu.edu.ua*. Зазвичай такий підхід використовують в маленьких мережах для спрощення роботи з сервісами цих мереж [Tom Yager] [7].

Іншим прикладом «приклеєною» записи може служити делегування зон всередині домену. Нехай ми створюємо два поддомена zone 1 і zone 2, сервери яких також розташовані в цих же піддоменів:

“$ORIGIN sumdu.edu.ua.

@ IN SOA gw.sumdu.edu.ua paul. ssu.edu.ua (

101 ; serial number

86400 ; refresh within a day

3600 ; retry every hour

3888000 ; expire after 45 days

99999999) ; default ttl is set to maximum value

IN NS gw.sumdu.edu.ua.

IN A 194.226.43.1;

gw IN A 194.226.43.1

; Glue address records.

zone1- gw.zone 1 IN A 194.226.43.2

zone2- gw.zone 2 IN A 194.226.43.3

; Subdomain delegation.

zone1 IN NS zone1-gw.zone1.sumdu.edu.ua.

zone2 IN NS zone2-gw.zone2.sumdu.edu.ua.”

Даний приклад - це розширення попереднього прикладу. Спочатку ми визначили IP-адреси серверів піддоменів в рамках поточного домену, а потім визначили їх в якості серверів відповідних зон нашого домену. При розбитті домену на зони не слід захоплюватися. Запам'ятовувати і набивати довгі імена також важко, як і числа IP-адреси.

Запис **Mail eXchanger**

Даний запис служить для визначення адреси поштового сервера. Запис MX може бути визначена як для всієї зони, так і для окремо взятої машини. Запис MX має такий вигляд [8]:

“[Name] [ttl] IN MX [prference] [host]” [8]

Поле name визначає ім'я машини або домену, на який може відправлятися пошта. Це може бути як повністю певне ім'я, так і часткове ім'я машини. Поле ttl зазвичай залишають порожнім. В поле preference вказується пріоритет поштового сервера, зазначеного останнім аргументом в поле даних MX -запісі. Запис MX використовується наступним чином:

· Поштовий агент, наприклад, програма sendmail, який вміє працювати зі службою доменних імен, отримує від сервера доменних імен у відповідь на свій запит список MX -записів даного домена;

В ньому він шукає записи, які дозволяють переслати пошту на машину, яка містить поштову скриньку адресата, або на машину, яка знає, як переслати пошту з даного поштовою адресою;

Використовуючи цю інформацію, sendmail пересилає пошту на знайдену машину. При передачі, в даному випадку, звичайно використовується протокол SMTP (Simple Mail Transfer Protocol) [4].

Типовими прикладами пересилання пошти є робота з поштою UUCP і BITNET. Пересилання стосується, як правило, йде з домену пошти, яка не є проблемою даного домену. У файлі настройки sendmail прописується ім'я відповідного шлюзу і на цьому все закінчується. Але для російських мереж тут можливі свої проблеми, які пов'язані з переходом від старої пошти UUCP, яка активно впроваджувалася мережею Relcom до системи роботи з такого самого протоколу, але вже через свій власний домен [4].

Наведемо приклад використання записів MX :

“$ORIGIN sumdu.edu.ua.

@ IN SOA gw.sumdu.edu.ua paul.ssu.edu.ua (

101 ; serial number

86400 ; refresh within a day

3600 ; retry every hour

3888000 ; expire after 45 days

99999999) ; default ttl is set to maximum value

IN NS gw.sumdu.edu.ua.

IN NS ns.relarn.ua.

IN NS polyn.ssu.edu.ua.

IN A 194.226.43.1

IN MX 10 gw.sumdu.edu.ua.

IN MX 20 ns.relarn.ua.

IN MX 30 relay.relarn.ua.

*.sumdu.edu.ua. IN MX 0 gw.sumdu.edu.ua.;

gw IN A 194.226.43.1

IN MX 0 gw

IN MX 10 ns.relarn.ua.

IN MX 20 relay.relarn.ua.”

В даному прикладі ми визначили декілька записів типу MX для типів поштової розсилки.

Записи в описі зони, відразу після A -записі, наступного за записом SOA, визначають шляхи надходження пошти в даний домен. Найвищий пріоритет тут має пряма відправка пошти на шлюз *gw.sumdu.edu.ua*. Якщо відправити пошту на цю машину не вдасться, то поштовий агент повинен спробувати шлях через поштовий сервер *ns.relarn.ua*. Якщо і цей шлях виявиться неможливим, то пошту можна спробувати відправити на пересильний пункт *relay.relarn.ua*. Такий порядок опитування поштових серверів визначається полем preference - чим менше значення поля, тим вище пріоритет сервера в запису MX.

У нашому домені, який ми використовуємо в якості прикладу, більшість поштових скриньок користувачів розташоване на машині *gw.sumdu.edu.ua*. Для цієї машини також вказані кілька записів MX. Найбільший пріоритет серед них має запис з ім'ям поштового сервера gw. Зверніть увагу на те, що дане ім'я не закінчується точкою. Це означає, що воно розширюється ім'ям поточної зони. Все ж інші імена серверів - це цілком конкретні доменні імена (fully-qualified).

Крім описаних MX -записів в нашому прикладі є ще одна, яка визначає поштовий шлюз для нашого домену - запис з ім'ям *«*.sumdu.edu.ua.”*. Даний запис введена для того, щоб визначити шлюз для будь-якої машини з нашого домену, якщо для неї не визначений шлюз в її власних MX -записів. Головним в цьому записі є використання символу «*», який позначає всі можливі імена машин і піддоменів даного домену.

*Запис призначення синоніма канонічного імені “*Canonical Name”

Запис CNAME визначає синоніми для реального доменного імені машини, яке визначено в запису типу A. Ім'я в запису типу A називають канонічним ім'ям машини. Формат запису CNAME можна визначити наступним чином [8]:

“[Nickname] [ttl] IN CNAME [host]” [8]

Поле nickname визначає синонім для канонічного імені, яке задається в поле host.

Запис CNAME найчастіше використовується для визначення інформаційних сервісів Internet, які встановлені на машині. Наступний приклад визначає синоніми, для комп'ютера, на якому встановлено сервери протоколів http і gopher :

“$ORIGIN polyn.ssu.edu.ua.

olga IN A 144.206.192.2

www IN CNAME olga.polyn.ssu.edu.ua.

gopher IN CNAME olga.polyn.ssu.edu.ua.”

Як і в попередніх прикладах команда $ORIGIN введена тільки для визначення поточної зони. Дві записи типу CNAME дозволяють організувати доступ до серверів, використовуючи імена розподілених інформаційних систем Internet, в рамках технологій яких працюють дані сервери. Саме такі синоніми і використовуються при зверненні до таких систем, як *Yahoo (www.Yahoo.Com)* або *Altavista (www.Altavista.Digital.Com).* Правда при зверненні до сервера протоколу http, який є базовим для системи World Wide Web, зміна імені машини, з якої здійснюють обслуговування, може бути

викликано багатьма причинами: перенаправленням запиту на інший сервер, використання віртуального сервера і тощо.

При використанні записів типу CNAME слід враховувати, що запис цього типу слід використовувати дуже акуратно. Деякі адміністратори рекомендують взагалі від неї відмовитися і якщо потрібно ще одне ім'я, то краще взагалі вказати ще одну A -запис для того ж самого IP-адреси. Зокрема запис типу MX може приєднуватися тільки до адресної записи. Якщо бути більш точним, то просто не всі поштові агенти здатні обробляти записи типу CNAME, тому що це вимагає додаткового звернення до сервера доменних імен. Наведемо приклад правильного і неправильного використання запису CNAME :

“$ORIGIN polyn. net. ssu.edu.ua.

olga IN A 144.206.192.2

IN MX 0 olga

IN MX 10 ns.polyn.ssu.edu.ua.

www IN CNAME olga.polyn.ssu.edu.ua.

gopher IN CNAME olga.polyn.ssu.edu.ua.”

В даному випадку записи типу CNAME вказані правильно, тому що не заважають переадресації пошти на машину olga. polyn. ssu.edu.ua. Але якщо їх поміняти місцями з записами типу MX, то, швидше за все пошта працювати не буде:

“$ORIGIN polyn.net.ssu.edu.ua.

olga IN A 144.206.192.2

www IN CNAME olga.polyn.ssu.edu.ua.

gopher IN CNAME olga.polyn.ssu.edu.ua.

IN MX 0 olga

IN MX 10 ns.polyn.ssu.edu.ua”

Це буде відбуватися з наступних причин. Поштовий агент для отримання адреси машини абонента поштового сервера у відповідь на свій запит отримає запис з бази даних сервера з ім'ям цієї машини. Якби це був

запис типу A, то по ній поштовий агент визначає IP-адресу машини-одержувача або машини шлюзу. Отримавши цю адресу агент організує взаємодію по протоколу SMTP і відправляє поштове повідомлення одержувачу або в чергу поштового агента шлюзу. Якщо ж він отримав в результаті звернення до сервера доменних імен запис типу CNAME, то ніякого IP-адреси в ній немає, а, отже, і організувати SMTP-з'єднання не можна.

Самі сервери доменних імен для визначення IP-адреси використовують кілька звернень. Коли отримано відповідь на запит, сервер аналізує тип отриманої записи, якщо це запис типу A, то IP-адреса передається системі resolver, якщо ж це запис типу CNAME, то сервер знову запитує віддалений сервер на предмет отримання нового запису, але вже по новому імені, яке він бере з поля host записи типу CNAME. Якщо в якості resolver використовується програма named, яка працює як resolver (команда slave в файлі named.boot), то в цьому випадку resolver вміє обробляти записи типу CNAME, але в більшості випадків, як у вже розглянутому нами випадку з електронною поштою, багато систем, які працюють в якості «мудрих» resolver'ов не вміють аналізувати записи типу CNAME, що і змушує використовувати замість них записи типу A. Крім того, останні мають ще й ту перевагу, що скорочується число запитів до сервера доменних імен, тому що IP-адреса, яка запитує сторона отримує з першої спроби.

В якості ще одного прикладу використання записів типу CNAME наведемо трасу отримання IP-адреси для доменного імені ftp.netscape.com:

“ftp.netscape.com. IN CNAME anonftp6.netscape.com.

anonftp6.netscape.com. IN CNAME ftp22.netscape.com.

ftp22.netscape.com. IN A 204.152.166.45”

Як видно з прикладу при пошуку IP-адреси для *ftp.netscape.com* локальний сервер доменних імен двічі отримує у відповідь на свій запит запис типу CNAME.

Записи типу “Pointer”

До сих пір ми описували записи, які найчастіше використовуються у файлі обслуговуючому «пряму» зону, тобто запити на отримання IP-адреси по доменному імені. Але існує, як ми з'ясували раніше, і запити іншого типу, коли необхідно по IP-адреси отримати доменне ім'я. Для реалізації відповідей на ці запити сервер веде опис «зворотної» зони, база даних яких складається із записів “Pointer”. Формат запису має наступний вигляд [8]:

“[Name] [ttl] IN PTR [host]” [8]

Поле name задає номер. Це не реальний IP-адресу машини, а ім'я в спеціальному домені .in-addr.arpa або в одній з його зон. Існує спеціальний формат запису імен цього домену. Так, наприклад, машина ns.sumdu.edu.ua, яка має адресу 176.108.235.100, повинна бути описана в домені in-addr.arpa як 100.235.108.176.in-addr.arpa, тобто адреса надається в зворотному порядку. Так як мова йде про доменної адресації, то розбиття на мережі не обовязкове. Таку систему доменних імен можна представити як на рис. 2.10:

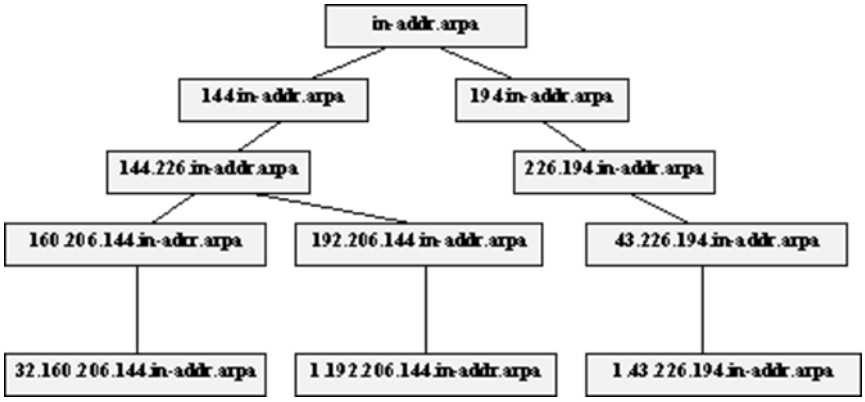


Рисунок 2.10 - Приклад структури частини домену .in-addr.arpa

Як видно з рисунку, в даному випадку не грає жодної ролі тип мережі або розбиття на підмережі. Згідно з правилами опису доменів і зон в цих доменах, роздільником в імені, яке визначає підлегле значення зони в домені, є тільки символ «.». Згідно з цим правилом підмережі мережі класу B (144.206.0.0) і мережа класу C (194.226.43.0) знаходяться на одному рівні ієрархії в системі домену .in-addr.arpa. Тому не треба комплексувати перед записами типу:

“$ORIGIN 194.in- addr.arpa.

1.43.226 IN PTR gw.sumdu.edu.ua.”

Однак слід визнати, що відповідність «зворотних» зон мереж - це загальна практика опису «зворотних» зон. Адже тут так само треба делегувати зони з «зворотного» домену. А робиться це, як правило, на основі розбиття мережі на підмережі або мережі.

Як приклад для нашої навчальної зони в файлі named.boot визначена «зворотна» зона 43.226.194.in-addr.arpa :

“@ IN SOA vega - gw. sumdu.edu.ua paul. ssu.edu.ua (

101 ; serial number

86400 ; refresh within a day

3600 ; retry every hour

3888000 ; expire after 45 days

99999999) ; default ttl is set to maximum value

IN NS gw.sumdu.edu.ua.

IN NS ns.relarn.ua. ;

1 IN PTR vega-gw.sumdu.edu.ua.

4 IN PTR dos1.sumdu.edu.ua.

5 IN PTR dos2.sumdu.edu.ua

2 IN PTR zone1-gw.zone1.sumdu.edu.ua.

3 IN PTR zone2-gw.zone2.sumdu.edu.ua.”

З цього прикладу видно, що для «зворотного» зони вказані ті ж записи опису зони, що і для «прямої». Крім того, «зворотний» зону зовсім не обов'язково розбивати відповідно до розподілу «прямий» зони. З точки зору домену .in-addr.arpa всі машини належать одній зоні - 43.226.194.in-addr.arpa.

Інша справа машина з IP-адреси 127.0.0.1. З точки зору домену .in-addr.arpa, вона належить іншій гілці ієрархії, відмінною від тієї, якої належать всі інші машини. Тому для домену 127.in-addr.arpa на будь-якій машині є окремий файл «зворотної» зони, хоча машина localhost, якій відповідає цей IP-адреса, зазвичай описується в «прямий» зоні домена разом з іншими машинами.

3. РЕАЛІЗАЦІЯ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ДНС СЕРВЕРОМ.

3.1. Створення БД проекту

Автоматизація роботи ДНС сервера пропонується зробити за допомогою розгортання кластера з 2х серверів на базі ОС Ubuntu 16.04.

Для цього нам знадобиться наступне ПО:

1. база даних MySQL;
2. сервіс Apache2;
3. сервіс Bind9 (named).

Розгортання серверів і сервісів буде покладено на відповідних адміністраторів організації.

Для роботи сервісу автоматизації ДНС сервера нам необхідно створити базу даних. Структура БД повинна містити в собі засоби для автоматизації авторизації користувачів в системі та таблицю сховища даних про записи доменних зон, розглянутих в теоретичній частині даної роботи.

Для реалізації вищеописаних вимог створений скрипт створення БД:

```
MariaDB [(none)]> CREATE TABLE domains (
id INT auto_increment,
name VARCHAR(255) NOT NULL,
master VARCHAR(128) DEFAULT NULL,
last_check INT DEFAULT NULL,
type VARCHAR(6) NOT NULL,
notified_serial INT DEFAULT NULL,
account VARCHAR(40) DEFAULT NULL,
primary key (id)
);
```

Рисунок 3.1 - Скрипт створення бази даних ч.1

Скрипт на рисунку 3.1 створює головну таблицю в БД, яка й буде тримати в собі дані з внесених домених імен.

```
MariaDB [(none)]> CREATE UNIQUE INDEX name_index ON domains(name);
MariaDB [(none)]> CREATE TABLE records (
id INT auto_increment,
domain_id INT DEFAULT NULL,
name VARCHAR(255) DEFAULT NULL,
type VARCHAR(6) DEFAULT NULL,
content VARCHAR(255) DEFAULT NULL,
ttl INT DEFAULT NULL,
prio INT DEFAULT NULL,
change_date INT DEFAULT NULL,
primary key(id)
);
```

Рисунок 3.2 - Скрипт створення БД ч.2

```
MariaDB [(none)]> CREATE INDEX rec_name_index ON records(name);
MariaDB [(none)]> CREATE INDEX nametype_index ON records(name,type);
MariaDB [(none)]> CREATE INDEX domain_id ON records(domain_id);
```

Рисунок 3.3 - Скрипт створення БД ч.3

```
MariaDB [(none)]> CREATE TABLE supermasters (
ip VARCHAR(25) NOT NULL,
nameserver VARCHAR(255) NOT NULL,
account VARCHAR(40) DEFAULT NULL
);
```

Рисунок 3.4 - Скрипт створення БД ч.4

Скрипти на рисунках 3.2 -3.4 продовжують процедуру створення БД. Цими скриптами створюється таблиці користувачів системи (підтримка різношарової авторизації) та таблиця для зберігання різних типів записів домених імен.

3.2. Створення конфігураційного файлу Apache2.

Підтримка управління і авторизації користувачів в системі буде реалізована за допомогою авторизації користувачів в системі через веб інтерфейс. Для реалізації даного функціоналу створюємо конфігураційний файл для ПО Apache 2:

```
<VirtualHost *:80>
        ServerName dns.sumdu.edu.ua
        Redirect permanent/https://dns.sumdu.edu.ua/
</VirtualHost>
```

Такми чином ми створюємо Віртуальний хост без рівня захисту, але забороняємо його використання, автоматично переадресовуємо всі запити на захищене з’єднання.

```
<VirtualHost *:443>
        ServerName dns.sumdu.edu.ua
        SSLEngine on
        SSLProtocol all -SSLv2
        SSLCertificateFile /etc/letsencrypt/live/dns.sumdu.edu.ua/cert.pem
        SSLCertificateKeyFile
/etc/letsencrypt/live/dns.sumdu.edu.ua/privkey.pem
        SSLCertificateChainFile
/etc/letsencrypt/live/dns.sumdu.edu.ua/chain.pem
        SSLUseStapling          on
        SSLStaplingResponderTimeout 5
        SSLStaplingReturnResponderErrors off
        Header always set Strict-Transport-Security "max-age=15768000;
includeSubDomains"
        DocumentRoot /srv/dnsui/public_html
        <Directory /srv/dnsui/public_html>
                AuthType Basic
                AuthName "DNS UI"
                AuthBasicProvider ldap
                AuthLDAPURL
ldaps://ldap.sumdu.edu.ua/dc=servers,dc=ua?uid?sub?(objectClass=inetOrgPerson)
                Require valid-user
                AllowOverride none
        </Directory>
        DirectoryIndex init.php
        FallbackResource /init.php
        AllowEncodedSlashes NoDecode
</VirtualHost>
```

Дана частина конфігурації використовується для створення віртуального хоста с захищеним типом з’єднання типу https. Шифрування пропонується проводити за допомогою безкоштовного ключа захисту letsencrypt. Основну директорію для скриптів пропонується використовувати за адресою /srv/dnsui/public_html.

3.3. Створення скриптів автоматизації управління.

Для проведення авторизації та робіт через веб інтерфейс створений скрипт внесення даних про ДНС зони до БД та роботи з ними на рівні користувачів (повний лістинг в додатку):

```
$testpath = !empty(basename($_SERVER ['REQUEST_URI'])) ? str_replace(basename($_SERVER ['REQUEST_URI']), $authdb, $_SERVER ['REQUEST_URI']) : '/'. $authdb;
$testurl = $_SERVER ['REQUEST_SCHEME']."://".$_SERVER ['HTTP_HOST'].$testpath;

global $errormsg, $blocklogin;
```

Ця частина скрипта відповідає за перевірку адреси, введеної в браузері, та підстановки її як перемінної в шляху роботи скрипта.

```
if (isset($_GET ['logout']) or isset($_POST ['logout'])) {
   logout();
   header("Location: index.php");
   exit(0);
}

if (!is_logged_in() and isset($_POST ['formname']) and $_POST ['formname'] === "loginform") {
   if (!try_login()) {
      $errormsg = "Не можемо вас автентифікувати\n";
   }
}

if (is_logged_in() and isset($_POST ['formname']) and $_POST ['formname'] === "changepwform") {
   if (get_sess_user() == $_POST ['username']) {
      if (!update_user(get_sess_userid(), is_adminuser(), $_POST ['password'])) {
         $errormsg = "Неможливо оновити пароль!\n";
      }
   } else {
      $errormsg = "Ви можете оновити лише свій пароль!".$_POST ['username'];
   }
}

?>
```

Частина скрипта: яка перевіряє авторизацію користувача та надає йому відповідні привілеї у відповідності до зазначених.

```
<!DOCTYPE html>
<html>
<head>
   <title>NSEdit!</title>
   <script type="text/javascript">
      var reader = new XMLHttpRequest();
      var checkFor = "<?php echo $testurl; ?>";
      reader.open('get', checkFor, true);
      reader.onreadystatechange = checkReadyState;
      function checkReadyState() {
         if (reader.readyState === 4) {
            //check to see whether request for the file failed or succeeded
            if ((reader.status == 200) || (reader.status == 0)) {
               alert('Your authdb is downloadable. Please secure your install');
            } else {
               return;
            }
         }

      }
      reader.send(null);
   </script>
   <link href="jquery-doc/shablon/base/all.css" rel="stylesheet" type="text/css"/>
   <link href="jtable/lib/shablon/metro/blue/jtable.min.css" rel="stylesheet"
type="text/css"/>
   <link href="css/base.css" rel="stylesheet" type="text/css"/>
   <?php if ($menutype === 'horizontal') { ?>
   <link href="css/horizontal-menu.css" rel="stylesheet" type="text/css"/>
   <?php } ?>
   <script src="jquery-doc/external/jquery/jquery.js"
type="text/javascript"></script>
   <script src="jquery-doc/ui/core.js" type="text/javascript"></script>
   <script src="jquery-doc/ui/widget.js" type="text/javascript"></script>
   <script src="jquery-doc/ui/mouse.js" type="text/javascript"></script>
   <script src="jquery-doc/ui/draggable.js" type="text/javascript"></script>
   <script src="jquery-doc/ui/position.js" type="text/javascript"></script>
   <script src="jquery-doc/ui/button.js" type="text/javascript"></script>
   <script src="jquery-doc/ui/resizable.js" type="text/javascript"></script>
   <script src="jquery-doc/ui/dialog.js" type="text/javascript"></script>
   <script src="jtable/lib/jquery.jtable.min.js" type="text/javascript"></script>
   <script src="js/addclear/addclear.js" type="text/javascript"></script>
```

```
</head>
```

Частина скрипта, яка відповідає за графічне відображення панелі керування в браузері та підєднує скрипти візуальних ефектів.

```
<?php
if (!is_logged_in()) {
?>
<body onload="document.getElementById('username').focus()">
<div class="loginblock">
   <div class="logo">
      <img src="<?php echo $logo ?>" alt="Logo"/>
   </div>
   <div class="login">
      <?php if (isset($errormsg)) {
         echo '<span style="color: red">'. $errormsg. '</span><br />';
      }
      ?>
      <form action="index.php" method="post">
         <table>
            <tr>
               <td class="label">Логін:</td>
               <td><input id="username" type="text" name="username"></td>
            </tr>
            <tr>
               <td class="label">Пароль:</td>
               <td><input type="password" name="password"></td>
            </tr>
            <?php
            if (isset($secret) && $secret) {
            ?>
            <?php
            }
            ?>
            <tr>
               <td></td>
               <td><input type="submit" name="submit" value="Ввійти!" <?php if
($blocklogin === TRUE) { echo "disabled"; }; ?>></td>
            </tr>
         </table>
         <input type="hidden" name="formname" value="loginform">
      </form>
   </div>
```

```
</div>
</body>
</html>

<?php
exit(0);
}
```

Функція підєднання до БД виконана таким чином:

```
function get_db() {
   global $authdb, $db;

   if (!isset($db)) {
      $db = new SQLite3($authdb, SQLITE3_OPEN_READWRITE);
      $db->exec('PRAGMA foreign_keys = 1');
   }
   return $db;
}
```

Функція роботи з Primary зонами описана наступним чином:

```
   $('#MasterZones').jtable({
      title: 'Master/Native Zones',
      paging: true,
      pageSize: 20,
      messages: {
         addNewRecord: 'Add new zone',
         editRecord: 'Edit zone',
         noDataAvailable: 'No zones found',
         deleteConfirmation: 'This zone will be deleted. Are you sure?'
      },
      toolbar: {
         hoverAnimation: true,
         hoverAnimationDuration: 60,
         hoverAnimationEasing: undefined,
         items: [
            <?php if (is_adminuser() or $allowzoneadd === TRUE) { ?>
            {
               icon: 'jtable/lib/shablon/metro/add.png',
               text: 'Import a new zone',
               click: function() {
                  $('#ImportZone').jtable('showCreateForm');
               }
            },
            {
               icon: 'jtable/lib/shablon/metro/add.png',
```

```
                text: 'Clone a zone',
                click: function() {
                    $('#CloneZone').jtable('showCreateForm');
                }
            },
            <?php } ?>
            ],
        },
        sorting: false,
        selecting: true,
        selectOnRowClick: true,
        selectionChanged: function (data) {
            var $selectedRows = $('#MasterZones').jtable('selectedRows');
            $selectedRows.each(function () {
                var zone = $(this).data('record');
                $('#MasterZones').jtable('openChildTable',
                    $(this).closest('tr'), {
                        title: 'Records in ' + zone.name,
                        messages: {
                            addNewRecord: 'Add to ' + zone.name,
                            noDataAvailable: 'No records for ' + zone.name
                        },
                        toolbar: {
                            items: [
                                {
                                    text: 'Search zone',
                                    click: function() {
                                        $("#searchzone").dialog({
                                            modal: true,
                                            title: "Search zone for ...",
                                            width: 'auto',
                                            buttons: {
                                                Search: function() {
                                                    $( this ).dialog( 'close' );
                                                    opentable.find('.jtable-title-
text').text(opentableTitle + " (filtered)");
                                                    opentable.jtable('load', {
                                                        label: $('#searchzone-label').val(),
                                                        type: $('#searchzone-type').val(),
                                                        content: $('#searchzone-content').val()
                                                    });
                                                },
                                                Reset: function() {
                                                    $('#searchzone-label').val('');
```

```
                                $('#searchzone-type').val('');
                                $('#searchzone-content').val('');
                                $( this ).dialog( 'close' );
                                opentable.find('.jtable-title-
text').text(opentableTitle);
                                opentable.jtable('load');
                                return false;
                            }
                        }
                    });
                }
            }
        ],
    },
    paging: true,
    sorting: true,
    pageSize: 20,
    openChildAsAccordion: true,
    actions: {
        listAction: 'zones.php?action=listrecords&zoneid=' + zone.id,
        createAction:       'zones.php?action=createrecord&zoneid='       +
zone.id,
        deleteAction:       'zones.php?action=deleterecord&zoneid='       +
zone.id,
        updateAction: 'zones.php?action=editrecord&zoneid=' + zone.id
    },
    fields: {
        domid: {
            create: true,
            type: 'hidden',
            defaultValue: zone.id
        },
        id: {
            key: true,
            type: 'hidden',
            create: false,
            edit: false,
            list: false
        },
        domain: {
            create: true,
            type: 'hidden',
            defaultValue: zone.name
        },
```

```
name: {
    title: 'Label',
    width: '7%',
    sorting: true,
    create: true,
    display: displayContent('name', zone.name),
    inputClass: 'name',
    listClass: 'name'
},
type: {
    title: 'Type',
    width: '2%',
    options: function() {
        zonename = new String(zone.name);
        if (zonename.match(/(\.in-addr|\.ip6)\.arpa/)) {
            return {
                'PTR': 'PTR',
                'NS': 'NS',
                'MX': 'MX',
                'TXT': 'TXT',
                'SOA': 'SOA',
                'A': 'A',
                'AAAA': 'AAAA',
                'CERT': 'CERT',
                'CNAME': 'CNAME',
                'ALIAS': 'ALIAS',
                'LOC': 'LOC',
                'NAPTR': 'NAPTR',
                'SPF': 'SPF',
                'SRV': 'SRV',
                'SSHFP': 'SSHFP',
                'TLSA': 'TLSA',
                'CAA': 'CAA',
                'DNAME': 'DNAME',
                'DS': 'DS',
                'SMIMEA': 'SMIMEA'
            };
        }
        return {
            'A': 'A',
            'AAAA': 'AAAA',
            'CERT': 'CERT',
            'CNAME': 'CNAME',
            'DNAME': 'DNAME',
```

```
            'ALIAS': 'ALIAS',
            'DS': 'DS',
            'LOC': 'LOC',
            'MX': 'MX',
            'NAPTR': 'NAPTR',
            'NS': 'NS',
            'PTR': 'PTR',
            'SOA': 'SOA',
            'SPF': 'SPF',
            'SRV': 'SRV',
            'SSHFP': 'SSHFP',
            'TLSA': 'TLSA',
            'CAA': 'CAA',
            'TXT': 'TXT',
            'SMIMEA': 'SMIMEA'
        };
    },
    display: displayContent('type'),
    create: true,
    inputClass: 'type',
    listClass: 'type'
},
content: {
    title: 'Content',
    width: '30%',
    create: true,
    sorting: true,
    display: displayContent('content'),
    inputClass: 'content',
    listClass: 'content'
},
ttl: {
    title: 'TTL',
    width: '2%',
    create: true,
    sorting: false,
    display: displayContent('ttl'),
    defaultValue: '<?php echo $defaults['ttl']; ?>',
    inputClass: 'ttl',
    listClass: 'ttl'
},
setptr: {
    title: 'Set PTR Record',
    width: '2%',
```

```
                        list: false,
                        create: true,
                        defaultValue: 'false',
                        inputClass: 'setptr',
                        listClass: 'setptr',
                        options: function() {
                            return {
                                '0': 'No',
                                '1': 'Yes',
                            };
                        },
                    },
                    disabled: {
                        title: 'Disabled',
                        width: '2%',
                        create: true,
                        sorting: false,
                        display: displayContent('disabled'),
                        defaultValue: '<?php echo $defaults['disabled'] ? 'No' : 'Yes'; ?>',
                        inputClass: 'disabled',
                        listClass: 'disabled',
                        options: function() {
                            return {
                                '0': 'No',
                                '1': 'Yes',
                            };
                        },
                    },
                }
            }, function (data) {
                opentable=data.childTable;
                opentableTitle=opentable.find('.jtable-title-text').text();
                data.childTable.jtable('load');
            });
        });
    },
    openChildAsAccordion: true,
    actions: {
        listAction: 'zones.php?action=list',
        <?php if (is_adminuser() or $allowzoneadd === TRUE) { ?>
        createAction: 'zones.php?action=create',
        deleteAction: 'zones.php?action=delete',
        <?php } ?>
```

```
    <?php if (is_adminuser()) { ?>
    updateAction: 'zones.php?action=update'
    <?php } ?>
},
fields: {
    id: {
        key: true,
        type: 'hidden'
    },
    name: {
        title: 'Domain',
        width: '8%',
        display: displayContent('name'),
        edit: false,
        inputClass: 'domain',
        listClass: 'domain'
    },
    dnssec: {
        title: 'DNSSEC',
        width: '3%',
        create: false,
        edit: false,
        display: displayDnssecIcon,
        listClass: 'dnssec'
    },
    <?php if (is_adminuser()) { ?>
    account: {
        title: 'Account',
        width: '8%',
        display: displayContent('account'),
        options: function(data) {
            return 'users.php?action=listoptions&e='+$epoch;
        },
        defaultValue: 'admin',
        inputClass: 'account',
        listClass: 'account'
    },
    <?php } ?>
    kind: {
        title: 'Type',
        width: '20%',
        display: displayContent('kind'),
        options: {'Native': 'Native', 'Master': 'Master'},
        defaultValue: '<?php echo $defaults['defaulttype']; ?>',
```

```
            edit: false,
            inputClass: 'kind',
            listClass: 'kind'
        },
        template: {
            title: 'Template',
            options: <?php echo json_encode(user_template_names()); ?>,
            list: false,
            create: true,
            edit: false,
            inputClass: 'template'
        },
        nameserver: {
            title: 'Nameservers',
            create: true,
            list: false,
            edit: false,
            input: function(data) {
                var $template = data.form.find('#Edit-template');
                var ns_form = '<?php foreach($defaults['ns'] as $ns) echo '<input type="text" name="nameserver[]" value="'.$ns.'" /><br />'; ?>';
                var $elem = $('<div id="nameservers">' + ns_form + '</div>');
                $template.change(function() {

$.get('zones.php?action=getformnameservers&template='+$template.val(), function(getdata) {
                    if (getdata != "") {
                            $("#nameservers").html(getdata);
                    } else {
                        $("#nameservers").html(ns_form);
                    }
                });
            });
            return $elem;
        },
        inputClass: 'nameserver nameserver1'
    },
    serial: {
        title: 'Serial',
        width: '10%',
        display: displayContent('serial'),
        create: false,
        edit: false,
        inputClass: 'serial',
```

```
            listClass: 'serial'
        },
        exportzone: {
            title: '',
            width: '1%',
            create: false,
            edit: false,
            display: displayExportIcon,
            listClass: 'exportzone'
        }
    }
});
```

Таким чином автоматизація управління корпоративним ДНС сервером завершена.

ВИСНОВКИ

1. Під час проведення дослідження було:
2. Проаналізовано існуючий ДНС сервер університету.
3. Проаналізовано технологію обробки даних та типи записів ДНС.
4. Проаналізовано сучасний стан роботи ДНС в Сумському державному університеті.
5. Розроблено прототип ДНС серверу університету.
6. Розроблено процес автоматизації управління ДНС сервером.

СПИСОК ЛІТЕРАТУРИ

1. “Информационная безопасность: защита и нападение.”, М. Бирюков А.А., ДМК Пресс, 2012 - 474стр. ISBN 5457427072, 9785457427075
2. «UNIX, В подлиннике», М., Магда Ю. С, БХВ-Петербург, 2006, 528 стр, ISBN 5941578245, 9785941578245.
3. «Unix и Linux: руководство системного администратора», М., Эви Немет, Гарт Снайдер, Трент Хейн, Бэн Уэйли, Litres, 2018, ISBN 5041399190, 9785041399191
4. Лукас М. FreeBSD. Подробное руководство, 2-е издание. - Пер. с англ. - СПб.: Символ-Плюс, 2009. - 864 с., ил., ISBN-10: 5-93286-126-6, ISBN-13: 978-5-93286-126-4, ISBN-13: 978-1-59327-151-0
5. Dynamic DNS & DHCP - http://www.bsdguides.org/guides/openbsd/networking/dynamic_dns_dhcp.php
6. Как правильно настроить DNS - http://www.nbsp.ua/articles/2005/12/14/kak_pravilno_nastroit_dns_chast1.html
7. The Arda.Homeunix.Net DNS & DHCP Setup - http://www.arda.homeunix.net/dnssetup.html
8. BIND9 Administrator Reference Manual - https://www.isc.org/files/Bv9.4ARM.pdf
9. Firewalling with OpenBSD's PF packet filter: Before we start - http://home.nuug.no/~peter/pf/en/preface.html
10. RFC 1034 — Domain Names — Concepts and Facilities - https://tools.ietf.org/html/rfc1034
11. RFC 1035 — Domain Names — Implementation and Specification - https://tools.ietf.org/html/rfc1035
12. RFC 1912 — Common DNS Operational and Configuration Errors - https://tools.ietf.org/html/rfc1912

13. RFC 1591 — Domain Name System Structure and Delegation - https://tools.ietf.org/html/rfc1591
14. RFC 1713 — Tools for DNS Debugging // https://tools.ietf.org/html/rfc1713
15. RFC 2606 — Reserved Top Level DNS Names - https://tools.ietf.org/html/rfc2606
16. «DNS and BIND: Help for System Administrators» , "O'Reilly Media, Inc.", 2006, Cricket Liu, Paul Albitz, 642 стр., ISBN 0596553404, 9780596553401
17. Brief History of the Domain Name System - https://cyber.harvard.edu/icann/pressingissues2000/briefingbook/dnshistory.html

ПЕРЕЛІК СКОРОЧЕНЬ

ДНС (DNS) – Domain Name service.

IP – інтернет адреса

БД – база даних;

ІБ – інформаційна безпека;

ІТ – інформаційні технології;

ОС – обчислювальна система;

ПБ – політика безпеки;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ДОДАТОК.

Додаток 1.

Лістинг коду автоматизації керування ДНС.

```
<?php

include_once('includes/config.inc.php');
include_once('includes/session.inc.php');
include_once('includes/misc.inc.php');

$testpath = !empty(basename($_SERVER ['REQUEST_URI'])) ?
str_replace(basename($_SERVER ['REQUEST_URI']), $authdb, $_SERVER
['REQUEST_URI']) : '/'. $authdb;
$testurl = $_SERVER ['REQUEST_SCHEME'].":".$_SERVER
['HTTP_HOST'].$testpath;

global $errormsg, $blocklogin;

if (isset($_GET ['logout']) or isset($_POST ['logout'])) {
    logout();
    header("Location: index.php");
    exit(0);
}

if (!is_logged_in() and isset($_POST ['formname']) and $_POST ['formname']
=== "loginform") {
    if (!try_login()) {
        $errormsg = "Error while trying to authenticate you\n";
    }
}

if (is_logged_in() and isset($_POST ['formname']) and $_POST ['formname'] ===
"changepwform") {
    if (get_sess_user() == $_POST ['username']) {
        if (!update_user(get_sess_userid(), is_adminuser(), $_POST
['password'])) {
            $errormsg = "Unable to update password!\n";
        }
    } else {
        $errormsg = "You can only update your own password!".$_POST
['username'];
    }
}

?>
<!DOCTYPE html>
<html>
<head>
    <title>NSEdit!</title>
    <script type="text/javascript">
        var reader = new XMLHttpRequest();
        var checkFor = "<?php echo $testurl; ?>";
        reader.open('get', checkFor, true);
        reader.onreadystatechange = checkReadyState;
        function checkReadyState() {
            if (reader.readyState === 4) {
                //check to see whether request for the file failed or
succeeded
                if ((reader.status == 200) || (reader.status == 0)) {
```

```
                    alert('Your authdb is downloadable. Please secure your
install');
                } else {
                    return;
                }
            }

        }
        reader.send(null);
    </script>
    <link href="jquery-doc/shablon/base/all.css" rel="stylesheet"
type="text/css"/>
    <link href="jtable/lib/shablon/metro/blue/jtable.min.css"
rel="stylesheet" type="text/css"/>
    <link href="css/base.css" rel="stylesheet" type="text/css"/>
    <?php if ($menutype === 'horizontal') { ?>
    <link href="css/horizontal-menu.css" rel="stylesheet" type="text/css"/>
    <?php } ?>
    <script src="jquery-doc/external/jquery/jquery.js"
type="text/javascript"></script>
    <script src="jquery-doc/ui/core.js" type="text/javascript"></script>
    <script src="jquery-doc/ui/widget.js" type="text/javascript"></script>
    <script src="jquery-doc/ui/mouse.js" type="text/javascript"></script>
    <script src="jquery-doc/ui/draggable.js" type="text/javascript"></script>
    <script src="jquery-doc/ui/position.js" type="text/javascript"></script>
    <script src="jquery-doc/ui/button.js" type="text/javascript"></script>
    <script src="jquery-doc/ui/resizable.js" type="text/javascript"></script>
    <script src="jquery-doc/ui/dialog.js" type="text/javascript"></script>
    <script src="jtable/lib/jquery.jtable.min.js"
type="text/javascript"></script>
    <script src="js/addclear/addclear.js" type="text/javascript"></script>
</head>

<?php
if (!is_logged_in()) {
?>
<body onload="document.getElementById('username').focus()">
<div class="loginblock">
    <div class="logo">
        <img src="<?php echo $logo ?>" alt="Logo"/>
    </div>
    <div class="login">
        <?php if (isset($errormsg)) {
            echo '<span style="color: red">'. $errormsg. '</span><br />';
        }
        ?>
        <form action="index.php" method="post">
            <table>
                <tr>
                    <td class="label">Username:</td>
                    <td><input id="username" type="text"
name="username"></td>
                </tr>
                <tr>
                    <td class="label">Password:</td>
                    <td><input type="password" name="password"></td>
                </tr>
                <?php
                if (isset($secret) && $secret) {
                ?>
                <tr>
                    <td class="label">Remember me:</td>
```

```
                    <td><input type="checkbox" name="autologin"
value="1"></td>
                </tr>
                <?php
                }
                ?>
                <tr>
                    <td></td>
                    <td><input type="submit" name="submit" value="Log me in!"
<?php if ($blocklogin === TRUE) { echo "disabled"; }; ?>></td>
                </tr>
            </table>
            <input type="hidden" name="formname" value="loginform">
        </form>
    </div>
</div>
</body>
</html>

<?php
exit(0);
}

if ($blocklogin === TRUE) {

       echo "<h2>There is an error in your config!</h2>";
       echo "<a href=\"index.php\">Refresh</a>";
       exit(0);
}

?>
<body>
<div id="wrap">
    <div id="dnssecinfo">
    </div>
    <div id="clearlogs" style="display: none;">
        Are you sure you want to clear the current logs? Maybe download them
        first<?php if($allowrotatelogs) { ?>, or use "Rotate logs" to save
        them on the server<?php } ?>?
    </div>
    <div id="rotatelogs" style="display: none;">
        Are you sure you want to rotate the current logs?
    </div>
    <div id="searchlogs" style="display: none; text-align: right;">
        <table border="0">
        <tr><td>User:</td><td><input type="text" id ="searchlogs-
user"><br></td></tr>
        <tr><td>Log Entry:</td><td><input type="text" id ="searchlogs-
entry"></td></tr>
        </table>
    </div>
    <div id="searchzone" style="display: none; text-align: right;">
        <table border="0">
        <tr><td>Label:</td><td><input type="text" id ="searchzone-
label"><br></td></tr>
        <tr><td>Type:</td><td style="text-align: left;"><select
id="searchzone-type">
            <option value=""></option>
            <option value="A">A</option>
            <option value="AAAA">AAAA</option>
            <option value="CERT">CERT</option>
            <option value="CNAME">CNAME</option>
            <option value="ALIAS">ALIAS</option>
```

```
            <option value="LOC">LOC</option>
            <option value="MX">MX</option>
            <option value="NAPTR">NAPTR</option>
            <option value="NS">NS</option>
            <option value="PTR">PTR</option>
            <option value="SOA">SOA</option>
            <option value="SPF">SPF</option>
            <option value="SRV">SRV</option>
            <option value="SSHFP">SSHFP</option>
            <option value="TLSA">TLSA</option>
            <option value="CAA">CAA</option>
            <option value="TXT">TXT</option>
            <option value="SMIMEA">SMIMEA</option>
        </select><br></td></tr>
        <tr><td>Content:</td><td><input type="text" id ="searchzone-
content"></td></tr>
        </table>
    </div>
    <div id="menu" class="jtable-main-container <?php if ($menutype ===
'horizontal') { ?>horizontal<?php } ?>">
        <div class="jtable-title menu-title">
            <div class="jtable-title-text">
                NSEdit!
            </div>
        </div>
        <ul>
            <li><a href="#" id="zoneadmin">Zones</a></li>
            <?php if (is_adminuser()) { ?>
                <li><a href="#" id="useradmin">Users</a></li>
                <li><a href="#" id="logadmin">Logs</a></li>
            <?php } ?>
            <li><a href="#" id="aboutme">About me</a></li>
            <li><a href="index.php?logout=1">Logout</a></li>
        </ul>
    </div>
    <?php if (isset($errormsg)) {
        echo '<span style="color: red">'. $errormsg. '</span><br />';
    }
    ?>
    <div id="zones">
        <?php if (is_adminuser() or $allowzoneadd === TRUE) { ?>
        <div style="display: none;" id="ImportZone"></div>
        <div style="display: none;" id="CloneZone"></div>
        <?php } ?>
        <div class="tables" id="MasterZones">
            <div class="searchbar" id="searchbar">
                <input type="text" id="domsearch" name="domsearch"
placeholder="Search...."/>
            </div>
        </div>
        <div class="tables" id="SlaveZones"></div>
    </div>
    <?php if (is_adminuser()) { ?>
    <div id="users">
        <div class="tables" id="Users"></div>
    </div>
    <div id="logs">
        <div class="tables" id="Logs"></div>
        <?php if($allowrotatelogs) { ?>
        <br>Log entries being viewed:
        <select id="logfile">
        <option value="">(Current logs)</option>
        <?php
```

```
            $logfiles=listrotatedlogs();
            if($logfiles !== FALSE) {
                foreach ($logfiles as $filename) {
                    echo '<option value="'. $filename. '">'.
str_replace(".json","",$filename). "</option>\n";
                }
            }
        ?></select>
        <?php } else { ?>
        <input type="hidden" id="logfile" value="">
        <?php } ?>
    </div>
    <?php } ?>

    <?php if (has_local_auth()) { ?>
    <div id="AboutMe">
        <div class="tables">
            <p>Hi <?php echo get_sess_user(); ?>. You can change your
password here.</p>

            <form action="index.php" method="POST">
                <table>
                    <tr>
                        <td class="label">Username:</td>
                        <td><input readonly value="<?php echo
get_sess_user(); ?>" id="username" type="text" name="username"></td>
                    </tr>
                    <tr>
                        <td class="label">Password:</td>
                        <td><input type="password" name="password"
id="changepw1"></td>
                    </tr>
                    <tr>
                        <td class="label">Password again:</td>
                        <td><input type="password" name="password2"
id="changepw2"></td>
                    </tr>
                    <tr>
                        <td></td>
                        <td><input type="submit" name="submit"
id="changepwsubmit" value="Change password!"></td>
                    </tr>
                </table>
                <input type="hidden" name="formname" value="changepwform">
                <input type="hidden" name="id" value="<?php echo
get_sess_userid(); ?>">
            </form>
        </div>
    </div>
    <?php } ?>
</div>
<script type="text/javascript">
window.csrf_token = '<?php echo CSRF_TOKEN ?>';

$(document).ready(function () {
    function csrfSafeMethod(method) {
        // these HTTP methods do not require CSRF protection
        return (/^(GET|HEAD|OPTIONS|TRACE)$/.test(method));
    }
    $.ajaxSetup({
        beforeSend: function(xhr, settings) {
            if (!csrfSafeMethod(settings.type) && !this.crossDomain) {
                xhr.setRequestHeader("X-CSRF-Token", window.csrf_token);
```

```
            }
        }
    });
});

function displayDnssecIcon(zone) {
    if (zone.record.dnssec == true) {
        var $img = $('<img class="clickme" src="img/lock.png" title="DNSSec
Info" />');
        $img.click(function () {
            $("#dnssecinfo").html("");
            $.each(zone.record.keyinfo, function (i, val) {
                if (val.dstxt) {

$("#dnssecinfo").append("<p><h2>"+val.keytype+"</h2><pre>"+val.dstxt+"</pre><
/p>");
                }
            });
            $("#dnssecinfo").dialog({
                modal: true,
                title: "DS-records for "+zone.record.name,
                width: 'auto',
                buttons: {
                    Ok: function() {
                        $(this).dialog("close");
                    }
                }
            });
        });
        return $img;
    } else {
        return '<img class="list" src="img/lock_open.png" title="DNSSec
Disabled" />';
    }
}

function displayExportIcon(zone) {
    var $img = $('<img class="list clickme" src="img/export.png"
title="Export zone" />');
    $img.click(function () {
        var $zexport =
$.getJSON("zones.php?zoneid="+zone.record.id+"&action=export", function(data)
{
            blob = new Blob( [data.Record.zone], { type: 'text/plain' });
            var dl = document.createElement('a');
            dl.addEventListener('click', function(ev) {
                dl.href = URL.createObjectURL(blob);
                dl.download = zone.record.name+'.txt';
            }, false);

            if (document.createEvent) {
                var event = document.createEvent("MouseEvents");
                event.initEvent("click", true, true);
                dl.dispatchEvent(event);
            }
        });
    });
    return $img;
}

function displayContent(fieldName, zone) {
    return function(data) {
        if (typeof(zone) != 'undefined') {
```

```
            var rexp = new RegExp("(.*)"+zone);
            var label = rexp.exec(data.record [fieldName]);
            var lspan = $('<span>').text(label [1]);
            var zspan = $('<span class="lightgrey">').text(zone);
            return lspan.add(zspan);
        } else {
            var text = data.record [fieldName];
            if (typeof data.record [fieldName] == 'boolean') {
                text == false ? text = 'No' : text = 'Yes';
            }
            return $('<span>').text(text);
        }
    }
}

function getEpoch() {
    return Math.round(+new Date()/1000);
}

$(document).ready(function () {
    var $epoch = getEpoch();

    $('#SlaveZones').jtable({
        title: 'Slave Zones',
        paging: true,
        pageSize: 20,
        sorting: false,
        messages: {
            addNewRecord: 'Add new slave zone',
            editRecord: 'Edit slave zone',
            noDataAvailable: 'No slave zones found',
            deleteConfirmation: 'This slave zone will be deleted. Are you
sure?'
        },
        openChildAsAccordion: true,
        actions: {
            listAction: 'zones.php?action=listslaves',
            updateAction: 'zones.php?action=update',
            <?php if (is_adminuser() or $allowzoneadd === TRUE) { ?>
            createAction: 'zones.php?action=create',
            deleteAction: 'zones.php?action=delete',
            <?php } ?>
        },
        fields: {
            id: {
                key: true,
                type: 'hidden'
            },
            name: {
                title: 'Domain',
                width: '8%',
                display: displayContent('name'),
                edit: false,
                inputClass: 'domain',
                listClass: 'domain'
            },
            dnssec: {
                title: 'DNSSEC',
                width: '3%',
                create: false,
                edit: false,
                display: displayDnssecIcon,
                listClass: 'dnssec'
```

```
            },
            <?php if (is_adminuser()) { ?>
            account: {
                title: 'Account',
                width: '8%',
                display: displayContent('account'),
                options: function(data) {
                    return 'users.php?action=listoptions&e='+$epoch;
                },
                defaultValue: 'admin',
                inputClass: 'account',
                listClass: 'account'
            },
            <?php } ?>
            kind: {
                create: true,
                type: 'hidden',
                list: false,
                defaultValue: 'Slave'
            },
            masters: {
                title: 'Masters',
                width: '20%',
                display: function(data) {
                    return $('<span>').text(data.record.masters.join('\n'));
                },
                input: function(data) {
                    var elem = $('<input type="text" name="masters">');
                    if (data && data.record) {
                        elem.attr('value', data.record.masters.join(','));
                    }
                    return elem;
                },
                inputClass: 'masters',
                listClass: 'masters'
            },
            serial: {
                title: 'Serial',
                width: '10%',
                display: displayContent('serial'),
                create: false,
                edit: false,
                inputClass: 'serial',
                listClass: 'serial'
            },
            records: {
                width: '5%',
                title: 'Records',
                paging: true,
                pageSize: 20,
                edit: false,
                create: false,
                display: function (zone) {
                    var $img = $('<img class="list" src="img/list.png"
title="Records" />');
                    $img.click(function () {
                        $('#SlaveZones').jtable('openChildTable',
                            $img.closest('tr'), {
                                title: 'Records in ' + zone.record.name,
                                openChildAsAccordion: true,
                                actions: {
                                    listAction:
'zones.php?action=listrecords&zoneid=' + zone.record.id
```

```
                            },
                            fields: {
                                name: {
                                    title: 'Label',
                                    width: '7%',
                                    display: displayContent('name'),
                                    listClass: 'name'
                                },
                                type: {
                                    title: 'Type',
                                    width: '2%',
                                    display: displayContent('type'),
                                    listClass: 'type'
                                },
                                content: {
                                    title: 'Content',
                                    width: '30%',
                                    display: displayContent('content'),
                                    listClass: 'content'
                                },
                                ttl: {
                                    title: 'TTL',
                                    width: '2%',
                                    display: displayContent('ttl'),
                                    listClass: 'ttl'
                                },
                                disabled: {
                                    title: 'Disabled',
                                    width: '2%',
                                    display: displayContent('disabled'),
                                    listClass: 'disabled'
                                }
                            }
                        }, function (data) {
                            data.childTable.jtable('load');
                        })
                    });
                    return $img;
                }
            },
            exportzone: {
                title: '',
                width: '1%',
                create: false,
                edit: false,
                display: displayExportIcon,
                listClass: 'exportzone'
            }
        }
    });
    $('#MasterZones').jtable({
        title: 'Master/Native Zones',
        paging: true,
        pageSize: 20,
        messages: {
            addNewRecord: 'Add new zone',
            editRecord: 'Edit zone',
            noDataAvailable: 'No zones found',
            deleteConfirmation: 'This zone will be deleted. Are you sure?'
        },
        toolbar: {
            hoverAnimation: true,
            hoverAnimationDuration: 60,
```

```
            hoverAnimationEasing: undefined,
            items:  [
                <?php if (is_adminuser() or $allowzoneadd === TRUE) { ?>
                {
                    icon: 'jtable/lib/shablon/metro/add.png',
                    text: 'Import a new zone',
                    click: function() {
                        $('#ImportZone').jtable('showCreateForm');
                    }
                },
                {
                    icon: 'jtable/lib/shablon/metro/add.png',
                    text: 'Clone a zone',
                    click: function() {
                        $('#CloneZone').jtable('showCreateForm');
                    }
                },
                <?php } ?>
                ],
        },
        sorting: false,
        selecting: true,
        selectOnRowClick: true,
        selectionChanged: function (data) {
            var $selectedRows = $('#MasterZones').jtable('selectedRows');
            $selectedRows.each(function () {
                var zone = $(this).data('record');
                $('#MasterZones').jtable('openChildTable',
                    $(this).closest('tr'), {
                        title: 'Records in ' + zone.name,
                        messages: {
                            addNewRecord: 'Add to ' + zone.name,
                            noDataAvailable: 'No records for ' + zone.name
                        },
                        toolbar: {
                            items:  [
                                {
                                    text: 'Search zone',
                                    click: function() {
                                        $("#searchzone").dialog({
                                            modal: true,
                                            title: "Search zone for...",
                                            width: 'auto',
                                            buttons: {
                                                Search: function() {
                                                    $(this).dialog('close');
                                                    opentable.find('.jtable-
title-text').text(opentableTitle + " (filtered)");
                                                    opentable.jtable('load',
{
                                                        label:
$('#searchzone-label').val(),
                                                        type: $('#searchzone-
type').val(),
                                                        content:
$('#searchzone-content').val()
                                                    });
                                                },
                                                Reset: function() {
                                                    $('#searchzone-
label').val('');
                                                    $('#searchzone-
type').val('');
```

```
                                                    $('#searchzone-
content').val('');
                                                    $(this).dialog('close');
                                                    opentable.find('.jtable-
title-text').text(opentableTitle);
                                                    opentable.jtable('load');
                                                    return false;
                                                }
                                            }
                                        });
                                    }
                                }
                            ],
                        },
                        paging: true,
                        sorting: true,
                        pageSize: 20,
                        openChildAsAccordion: true,
                        actions: {
                            listAction:
'zones.php?action=listrecords&zoneid=' + zone.id,
                            createAction:
'zones.php?action=createrecord&zoneid=' + zone.id,
                            deleteAction:
'zones.php?action=deleterecord&zoneid=' + zone.id,
                            updateAction:
'zones.php?action=editrecord&zoneid=' + zone.id
                        },
                        fields: {
                            domid: {
                                create: true,
                                type: 'hidden',
                                defaultValue: zone.id
                            },
                            id: {
                                key: true,
                                type: 'hidden',
                                create: false,
                                edit: false,
                                list: false
                            },
                            domain: {
                                create: true,
                                type: 'hidden',
                                defaultValue: zone.name
                            },
                            name: {
                                title: 'Label',
                                width: '7%',
                                sorting: true,
                                create: true,
                                display: displayContent('name', zone.name),
                                inputClass: 'name',
                                listClass: 'name'
                            },
                            type: {
                                title: 'Type',
                                width: '2%',
                                options: function() {
                                    zonename = new String(zone.name);
                                    if (zonename.match(/(\.in-
addr|\.ip6)\.arpa/)) {
                                        return {
```

```
                    'PTR': 'PTR',
                    'NS': 'NS',
                    'MX': 'MX',
                    'TXT': 'TXT',
                    'SOA': 'SOA',
                    'A': 'A',
                    'AAAA': 'AAAA',
                    'CERT': 'CERT',
                    'CNAME': 'CNAME',
                    'ALIAS': 'ALIAS',
                    'LOC': 'LOC',
                    'NAPTR': 'NAPTR',
                    'SPF': 'SPF',
                    'SRV': 'SRV',
                    'SSHFP': 'SSHFP',
                    'TLSA': 'TLSA',
                    'CAA': 'CAA',
                    'DNAME': 'DNAME',
                    'DS': 'DS',
                    'SMIMEA': 'SMIMEA'
                };
            }
            return {
                'A': 'A',
                'AAAA': 'AAAA',
                'CERT': 'CERT',
                'CNAME': 'CNAME',
                'DNAME': 'DNAME',
                'ALIAS': 'ALIAS',
                'DS': 'DS',
                'LOC': 'LOC',
                'MX': 'MX',
                'NAPTR': 'NAPTR',
                'NS': 'NS',
                'PTR': 'PTR',
                'SOA': 'SOA',
                'SPF': 'SPF',
                'SRV': 'SRV',
                'SSHFP': 'SSHFP',
                'TLSA': 'TLSA',
                'CAA': 'CAA',
                'TXT': 'TXT',
                'SMIMEA': 'SMIMEA'
            };
        },
        display: displayContent('type'),
        create: true,
        inputClass: 'type',
        listClass: 'type'
    },
    content: {
        title: 'Content',
        width: '30%',
        create: true,
        sorting: true,
        display: displayContent('content'),
        inputClass: 'content',
        listClass: 'content'
    },
    ttl: {
        title: 'TTL',
        width: '2%',
        create: true,
```

```
                                sorting: false,
                                display: displayContent('ttl'),
                                defaultValue: '<?php echo $defaults ['ttl'];
?>',
                                inputClass: 'ttl',
                                listClass: 'ttl'
                            },
                            setptr: {
                                title: 'Set PTR Record',
                                width: '2%',
                                list: false,
                                create: true,
                                defaultValue: 'false',
                                inputClass: 'setptr',
                                listClass: 'setptr',
                                options: function() {
                                    return {
                                        '0': 'No',
                                        '1': 'Yes',
                                    };
                                },
                            },
                            disabled: {
                                title: 'Disabled',
                                width: '2%',
                                create: true,
                                sorting: false,
                                display: displayContent('disabled'),
                                defaultValue: '<?php echo $defaults
['disabled'] ? 'No' : 'Yes'; ?>',
                                inputClass: 'disabled',
                                listClass: 'disabled',
                                options: function() {
                                    return {
                                        '0': 'No',
                                        '1': 'Yes',
                                    };
                                },
                            },
                        }
                    }, function (data) {
                        opentable=data.childTable;
                        opentableTitle=opentable.find('.jtable-title-
text').text();
                        data.childTable.jtable('load');
                    });
            });
        },
        openChildAsAccordion: true,
        actions: {
            listAction: 'zones.php?action=list',
            <?php if (is_adminuser() or $allowzoneadd === TRUE) { ?>
            createAction: 'zones.php?action=create',
            deleteAction: 'zones.php?action=delete',
            <?php } ?>
            <?php if (is_adminuser()) { ?>
            updateAction: 'zones.php?action=update'
            <?php } ?>
        },
        fields: {
            id: {
                key: true,
                type: 'hidden'
```

```
            },
            name: {
                title: 'Domain',
                width: '8%',
                display: displayContent('name'),
                edit: false,
                inputClass: 'domain',
                listClass: 'domain'
            },
            dnssec: {
                title: 'DNSSEC',
                width: '3%',
                create: false,
                edit: false,
                display: displayDnssecIcon,
                listClass: 'dnssec'
            },
            <?php if (is_adminuser()) { ?>
            account: {
                title: 'Account',
                width: '8%',
                display: displayContent('account'),
                options: function(data) {
                    return 'users.php?action=listoptions&e='+$epoch;
                },
                defaultValue: 'admin',
                inputClass: 'account',
                listClass: 'account'
            },
            <?php } ?>
            kind: {
                title: 'Type',
                width: '20%',
                display: displayContent('kind'),
                options: {'Native': 'Native', 'Master': 'Master'},
                defaultValue: '<?php echo $defaults ['defaulttype']; ?>',
                edit: false,
                inputClass: 'kind',
                listClass: 'kind'
            },
            template: {
                title: 'Template',
                options: <?php echo json_encode(user_template_names()); ?>,
                list: false,
                create: true,
                edit: false,
                inputClass: 'template'
            },
            nameserver: {
                title: 'Nameservers',
                create: true,
                list: false,
                edit: false,
                input: function(data) {
                    var $template = data.form.find('#Edit-template');
                    var ns_form = '<?php foreach($defaults ['ns'] as $ns)
echo '<input type="text" name="nameserver []" value="'.$ns.'" /><br />'; ?>';
                    var $elem = $('<div id="nameservers">' + ns_form +
'</div>');
                    $template.change(function() {

$.get('zones.php?action=getformnameservers&template='+$template.val(),
function(getdata) {
```

```
                        if (getdata != "") {
                           $("#nameservers").html(getdata);
                        } else {
                            $("#nameservers").html(ns_form);
                        }
                    });
                });
                return $elem;
            },
            inputClass: 'nameserver nameserver1'
        },
        serial: {
            title: 'Serial',
            width: '10%',
            display: displayContent('serial'),
            create: false,
            edit: false,
            inputClass: 'serial',
            listClass: 'serial'
        },
        exportzone: {
            title: '',
            width: '1%',
            create: false,
            edit: false,
            display: displayExportIcon,
            listClass: 'exportzone'
        }
    }
});
$('#ImportZone').jtable({
    title: 'Import zone',
    actions: {
        createAction: 'zones.php?action=create'
    },
    fields: {
        id: {
            key: true,
            type: 'hidden'
        },
        name: {
            title: 'Domain',
            inputClass: 'domain'
        },
        <?php if (is_adminuser()) { ?>
        account: {
            title: 'Account',
            options: function(data) {
                return 'users.php?action=listoptions&e='+$epoch;
            },
            defaultValue: 'admin',
            inputClass: 'account'
        },
        <?php } ?>
        kind: {
            title: 'Type',
            options: {'Native': 'Native', 'Master': 'Master'},
            defaultValue: '<?php echo $defaults ['defaulttype']; ?>',
            edit: false,
            inputClass: 'type'
        },
        zone: {
            title: 'Zonedata',
```

```
                type: 'textarea',
                inputClass: 'zonedata'
            },
            owns: {
                title: 'Overwrite Nameservers',
                type: 'checkbox',
                values: {'0': 'No', '1': 'Yes'},
                defaultValue: 1,
                inputClass: 'overwrite_nameserver'
            },
            nameserver: {
                title: 'Nameservers',
                create: true,
                list: false,
                edit: false,
                input: function(data) {
                    var ns_form = '<?php foreach($defaults ['ns'] as $ns)
echo '<input type="text" name="nameserver []" value="'.$ns.'" /><br />'; ?>';
                    var $elem = $('<div id="nameservers">' + ns_form +
'</div>');
                    return $elem;
                },
                inputClass: 'nameserver nameserver1'
            },
        },
        recordAdded: function() {
            $("#MasterZones").jtable('load');
            $("#SlaveZones").jtable('load');
        }

    });

    $('#CloneZone').jtable({
        title: 'Clone zone',
        actions: {
            createAction: 'zones.php?action=clone'
        },
        fields: {
            id: {
                key: true,
                type: 'hidden'
            },
            sourcename: {
                title: 'Source domain',
                options: function(data) {
                    return 'zones.php?action=formzonelist&e='+$epoch;
                },
                inputClass: 'sourcename'
            },
            destname: {
                title: 'Domain',
                inputClass: 'destname'
            },
            account: {
                title: 'Account',
                options: function(data) {
                    return 'users.php?action=listoptions&e='+$epoch;
                },
                defaultValue: 'admin',
                inputClass: 'account'
            },
            kind: {
                title: 'Type',
```

```
            options: {'Native': 'Native', 'Master': 'Master'},
            defaultValue: '<?php echo $defaults ['defaulttype']; ?>',
            edit: false,
            inputClass: 'type'
        },
    },
    recordAdded: function() {
        $("#MasterZones").jtable('load');
        $("#SlaveZones").jtable('load');
    }

});

$('#domsearch').addClear({
    onClear: function() { $('#MasterZones').jtable('load'); }
});

function searchDoms() {
    $('#MasterZones').jtable('load', {
        domsearch: $('#domsearch').val()
    });
    $('#SlaveZones').jtable('load', {
        domsearch: $('#domsearch').val()
    });
}

stimer = 0;

$('#changepw1, #changepw2').on('input', function(e) {
    if ($('#changepw1').val() != $('#changepw2').val()) {
        $('#changepwsubmit').prop("disabled",true);
    } else {
        $('#changepwsubmit').prop("disabled",false);
    }
});

$('#domsearch').on('input', function (e) {
    e.preventDefault();
    clearTimeout(stimer);
    stimer = setTimeout(searchDoms, 400);
});

<?php if (is_adminuser()) { ?>
$('#logs').hide();
$('#Users').hide();
$('#AboutMe').hide();
$('#aboutme').click(function () {
    $('#logs').hide();
    $('#Users').hide();
    $('#MasterZones').hide();
    $('#SlaveZones').hide();
    $('#AboutMe').show();
});
$('#useradmin').click(function () {
    $('#logs').hide();
    $('#MasterZones').hide();
    $('#SlaveZones').hide();
    $('#AboutMe').hide();
    $('#Users').jtable('load');
    $('#Users').show();
});
$('#zoneadmin').click(function () {
    $('#logs').hide();
```

```
    $('#Users').hide();
    $('#AboutMe').hide();
    $('#MasterZones').show();
    $('#SlaveZones').show();
});
$('#logadmin').click(function () {
    $('#Users').hide();
    $('#AboutMe').hide();
    $('#MasterZones').hide();
    $('#SlaveZones').hide();
    $('#Logs').jtable('load', {
        logfile: $('#logfile').val()
    });
    $('#logs').show();
});
$('#Users').jtable({
    title: 'Users',
    paging: true,
    pageSize: 20,
    sorting: false,
    actions: {
        listAction: 'users.php?action=list',
        createAction: 'users.php?action=create',
        deleteAction: 'users.php?action=delete',
        updateAction: 'users.php?action=update'
    },
    messages: {
        addNewRecord: 'Add new user',
        deleteConfirmation: 'This user will be deleted. Are you sure?'
    },
    fields: {
        id: {
            key: true,
            type: 'hidden'
        },
        emailaddress: {
            title: 'User',
            display: displayContent('emailaddress'),
            inputClass: 'emailaddress',
            edit: false,
            listClass: 'emailaddress'
        },
        password: {
            title: 'Password',
            type: 'password',
            list: false,
            inputClass: 'password',
        },
        isadmin: {
            title: 'Admin',
            type: 'checkbox',
            values: {'0': 'No', '1': 'Yes'},
            inputClass: 'isadmin',
            listClass: 'isadmin'
        }
    },
    recordAdded: function() {
        $epoch = getEpoch();
        $("#MasterZones").jtable('reload');
        $("#SlaveZones").jtable('reload');
    }
});
```

```
    $('#Logs').jtable({
        title: 'Logs',
        paging: true,
        pageSize: 20,
        sorting: false,
        actions: {
            listAction: 'logs.php?action=list'
        },
        messages: {
            deleteConfirmation: 'This entry will be deleted. Are you sure?'
        },
        toolbar: {
            hoverAnimation: true,
            hoverAnimationDuration: 60,
            hoverAnimationEasing: undefined,
            items:  [
                {
                    text: 'Search logs',
                    click: function() {
                        $("#searchlogs").dialog({
                            modal: true,
                            title: "Search logs for...",
                            width: 'auto',
                            buttons: {
                                Search: function() {
                                    $(this).dialog('close');
                                    $('#Logs').find('.jtable-title-
text').text('Logs (filtered)');
                                    $('#Logs').jtable('load', {
                                        logfile: $('#logfile').val(),
                                        user: $('#searchlogs-user').val(),
                                        entry: $('#searchlogs-entry').val()
                                    });
                                },
                                Reset: function() {
                                    $('#searchlogs-user').val('');
                                    $('#searchlogs-entry').val('');
                                    $(this).dialog('close');
                                    $('#Logs').find('.jtable-title-
text').text('Logs');
                                    $('#Logs').jtable('load', {
                                        logfile: $('#logfile').val()
                                    });
                                    return false;
                                }
                            }
                        });
                    }
                },
                <?php if($allowrotatelogs === TRUE) { ?>
                {
                    text: 'Rotate logs',
                    click: function() {
                        $("#rotatelogs").dialog({
                            modal: true,
                            title: "Rotate logs",
                            width: 'auto',
                            buttons: {
                                Ok: function() {
                                    $.get("logs.php?action=rotate");
                                    $(this).dialog("close");
                                    $('#logfile').val('');
                                    $('#Logs').jtable('load');
```

```
                                },
                                Cancel: function() {
                                    $(this).dialog("close");
                                    return false;
                                }
                            }
                        });
                    }
                },
                <?php } ?>
                <?php if($allowclearlogs === TRUE) { ?>
                {
                    icon: 'img/delete_inverted.png',
                    text: 'Clear logs',
                    click: function() {
                        $("#clearlogs").dialog({
                            modal: true,
                            title: "Clear all logs",
                            width: 'auto',
                            buttons: {
                                Ok: function() {
                                    $.get("logs.php?action=clear");
                                    $(this).dialog("close");
                                    $('#logfile').val('');
                                    $('#Logs').jtable('load');
                                },
                                Cancel: function() {
                                    $(this).dialog("close");
                                    return false;
                                }
                            }
                        });
                    }
                },
                <?php } ?>
                {
                    icon: 'img/export.png',
                    text: 'Download logs',
                    click: function () {
                        var $zexport =
$.get("logs.php?action=export&logfile=" + $('#logfile').val(), function(data)
{
                            console.log(data);
                            blob = new Blob( [data], { type: 'text/plain' });
                            var dl = document.createElement('a');
                            dl.addEventListener('click', function(ev) {
                                dl.href = URL.createObjectURL(blob);
                                dl.download = $('#logfile').val() == "" ?
'nseditlogs.txt':$('#logfile').val() + ".txt";
                            }, false);

                            if (document.createEvent) {
                                var event =
document.createEvent("MouseEvents");
                                event.initEvent("click", true, true);
                                dl.dispatchEvent(event);
                            }
                        });
                    }
                }
                ],
        },
        fields: {
```

```
            id: {
                title: 'key',
                key: true,
                type: 'hidden'
            },
            user: {
                title: 'User',
                width: '10%',
                display: displayContent('user'),
            },
            log: {
                title: 'Log',
                width: '80%',
                display: displayContent('log'),
            },
            timestamp: {
                title: 'Timestamp',
                width: '10%',
                display: displayContent('timestamp')
            }
        }
    });

    $('#logfile').change(function () {
        $('#Logs').jtable('load', {
            logfile: $('#logfile').val(),
            user: $('#searchlogs-user').val(),
            entry: $('#searchlogs-entry').val()
        });
    });
    <?php } else { ?>
    $('#AboutMe').hide();
    $('#aboutme').click(function () {
        $('#MasterZones').hide();
        $('#SlaveZones').hide();
        $('#AboutMe').show();
    });
    $('#zoneadmin').click(function () {
        $('#AboutMe').hide();
        $('#MasterZones').show();
        $('#SlaveZones').show();
    });

    <?php } ?>
    $('#MasterZones').jtable('load');
    $('#SlaveZones').jtable('load');
});
</script>
</body>
</html>
```