

Brought to you by

Infoblox

DNS Security

for
dummies[®]

A Wiley Brand

Infoblox Special Edition



Improve application
and service availability

—

Protect users and data
from malware

—

Optimize your
security operations

Joshua M. Kuo

Robert Nagy

Cricket Liu

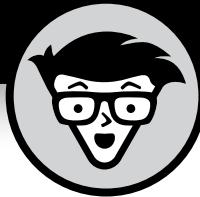
About Infoblox

Network landscapes are rapidly evolving, driven by trends in security, virtualization, cloud, IPv6 adoption, and the Internet of Things (IoT). Conventional network management solutions are manual, fragmented, and vulnerable to attack. They're no longer able to keep pace with the exponential growth of devices, IP traffic, and sophisticated security threats.

For more than 17 years, Infoblox has been a market leader in DNS, DHCP, and IP address management (DDI). They provide control and security to keep pace with the growing network landscapes. They're empowering thousands of organizations to pursue the big initiatives that drive business success today — next-generation data centers, security, compliance, and digital transformation — and to increase efficiency and visibility, reduce risk, and improve customer experience.

Infoblox put their extensive experience in DDI technology to work to create the industry's first Actionable Network Intelligence Platform. It goes beyond DDI to enable users to harness insights derived from the rivers of data moving through networks to enhance all aspects of network management, security, agility, and cost control. Infoblox Actionable Network Intelligence addresses business needs.

Infoblox delivers products and solutions to more than 8,500 enterprises, government agencies, and service providers in more than 25 countries around the world — including 7 of the top 10 aerospace and defense companies, 7 of the top 10 telecommunications providers, 8 of the top 10 retailers, 8 of the top 10 major banks, and 9 of the top 10 automakers.



DNS Security

Infoblox Special Edition

**by Joshua M. Kuo, Robert Nagy,
and Cricket Liu**

FOREWORD BY Cricket Liu

for
dummies[®]

A Wiley Brand

DNS Security For Dummies®, Infoblox Special Edition

Published by

John Wiley & Sons, Inc.

111 River St.

Hoboken, NJ 07030-5774

www.wiley.com

Copyright © 2018 by John Wiley & Sons, Inc., Hoboken, New Jersey

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Trademarks: Wiley, For Dummies, the Dummies Man logo, The Dummies Way, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. Infoblox and the Infoblox logo are trademarks or registered trademarks of Infoblox, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc., is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

ISBN 978-1-119-43731-4 (pbk); ISBN 978-1-119-43728-4 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Editor: Jennifer Bingham

Acquisitions Editor: Amy Fandrei

Editorial Manager: Rev Mengle

Business Development

Representative: Karen Hattan

Production Editor: G. Vasanth Koilraj

Foreword

Paul Albitz and I wrote the first edition of *DNS and BIND* way back in 1992. (Well, actually, we started it about 14 months before that, so in 1991.) Back then, DNS security wasn't a thing. BIND 4.8.3, the version of the BIND DNS server that was current when we wrote that first edition, had the following security features:

- » It wouldn't accept a response it received from a DNS server it hadn't queried (playfully dubbed a Martian response).
- » It stamped a random, 16-bit number, called a Message ID, into each outbound query it sent to a remote DNS server, and when it received a response from that DNS server, it made sure the Message ID matched.
- » That's it.

Over the years, DNS — both the protocol and the servers — became the target of a variety of attacks, including the Lion worm, a cache poisoning attack on www.internic.net, social engineering attacks against registrar accounts, and distributed denial of service attacks on DNS servers. And so the DNS community developed new mechanisms to combat these attacks including access controls on queries, dynamic updates, and zone transfers; DNS security extensions; response policy zones; and response rate limiting.

Unfortunately, I've been remiss in keeping *DNS and BIND* up to date on all these new developments. (I blame that mostly on the demands of raising a family and working a full-time job, and only partly on being daunted and disheartened by the thought of all of the research and writing involved.)

The good news for you, dear reader, and for me, is that the little book you're holding will help make up for my negligence by providing you with an overview on the new security mechanisms in DNS. Rob and Josh know their stuff: They've been developing and delivering courses on DNS and DNS security for my

company, Infoblox, for years. They'll give you an overview of the most important DNS security technologies and advice on when you should apply them. (They even generously gave me credit as a co-author for providing the outline and a little help here and there.) And with a little further research and effort on your part, that could lead you to building more secure and more robust DNS infrastructure!

—Cricket Liu, Chief DNS Architect and Senior Fellow at Infoblox

Introduction

Everybody uses the Internet. The Internet is so intrinsic to modern life that everyone takes it for granted. However, a worldwide network of computing power doesn't just work on its own. Over the relatively short life of the Internet, many sophisticated technologies, such as DNS, have grown to make the convenience that we've come to expect from the Internet possible.

The Internet is the target of attacks by unethical people. These people might just want to cause mischief, or they might want to extort you for large amounts of money. Either way, it's important to know how to protect yourself and your electronic property.

We have a passion for just that kind of protection. That's why we wrote this book: We want to give you a solid understanding of how the most common types of attacks work. Moreover, we want to offer the know-how to keep you safe.

About This Book

You've probably picked up this book because you're confident enough to admit that you don't know as much about DNS security and are clever enough to look for more knowledge. We don't think you'll be disappointed. In this brief volume, we offer a primer of many of the common terms you'll run into, high-level descriptions of the threats you face, and practical solutions that you can implement right away.

Like all titles in the *For Dummies* series, this book features easy-access organization. At the beginning of each chapter, you'll find a summary of the topics covered, which makes it easy to flip through and find just the information you're looking for. Don't miss the final chapter featuring ten easy-to-scan techniques for improving your DNS security.

Foolish Assumptions

DNS security isn't exactly a cocktail party conversation topic, so we assume that readers of this book have a vested interest in keeping corporate websites functioning and secure. However, we tried to write this book so that all people who pick up a copy can learn something new and interesting that deepens their understanding of Internet security.

You can't write a book like this without making a few assumptions, though. For this book, we assume that you're an experienced user of the Internet. We define most of our terms, but we do assume you understand the basics of networking like *server*, *client*, and *IP address*.

Icons Used in This Book

Throughout this book, we occasionally use special icons to call attention to important information. Here's what to expect:



REMEMBER

Whenever you see a Remember icon, make a mental note about what you're reading, because this information may turn up again in this book.



TECHNICAL STUFF

The Technical Stuff icon marks extra-technical reading. You can skip it if you want or come back to it later.



TIP

The Tip icon points out information that's handy to know



WARNING

This icon points out some dangers to be on the lookout for.

Beyond the Book

If any of these topics in this book has you scratching your head, go ahead and read on anyway, then bring your questions to the Support menu on <https://infoblox.com>. They would love to explain more.

IN THIS CHAPTER

- » Learning what DNS is and where it came from
- » Discovering how DNS works
- » Reviewing types of resource records
- » Understanding queries, recursion, and iteration

Chapter 1

DNS 101

The amount of data that the Internet contains is growing at an astronomical pace. A single computer doesn't hold it all, of course; this much data must be distributed across countless computers all over the world. Even so, with an Internet connection, you can navigate to any file on the Internet as easily as you find a file on your own hard drive.

This amazing capability comes from the Domain Name System, or DNS. DNS is the tool that your browser uses to quickly find a file that might be stored in a computer anywhere on earth.

What Is DNS?

Although phone books are quickly going out of style, many people still remember what they are. DNS works very much like a phone book in that it helps turn names (URLs for resources on the Internet) into numbers (IP addresses of the computer that contains the resource).

Perhaps a more modern example would be to compare DNS to the contacts in a smartphone. Not many folks nowadays memorize each other's phone numbers; we rely on the contact application in our phones to translate names or faces to phone numbers. In essence, that is what DNS does.

DNS History

DNS stands for *Domain Name System* and is an Internet protocol that converts human-readable names to IP addresses, changes IP addresses back to names, and provides easy-to-remember names for many Internet-based services, such as email.

At the dawning of the Internet, or as it was known back then, the ARPANET (Advanced Research Projects Agency Network), very few people and machines were actually online. Each computer using the Internet had an IP address, but since there were so few IP addresses, memorizing them wasn't a big deal.



TECHNICAL
STUFF

As the number of machines quickly grew, people thought it would be a good idea to use more human-friendly names. Instead of remembering a computer's IP address, such as 128.171.32.45, ARPANET, users could enter names such as GOPHER-HAWAII. A single text file named HOSTS.TXT served as a name-to-address map. The Stanford Research Institute (then a part of ARPANET) manually maintained the file, also known as *the hosts file*, in a single place, and distributed it to ARPANET users.

Back then, if you wanted to translate a name to an IP address, you needed to download the latest copy of the hosts file. Likewise, if you wanted to be known by the other parts of ARPANET by name, you needed to contact the maintainer of the hosts file and add yourself to the list.

This centralized system quickly proved unscalable. Computer scientist and Internet pioneer Paul Mockapetris began work writing a standards document to define a replacement for host files. He took his proposed standard to the Internet Engineering Task Force (IETF), which still today produces standards documents that define how Internet protocols should operate and interoperate.

In 1983, Mockapetris published the first standards documents in the IETF that would become the basis for the DNS. His proposal called for a decentralized, distributed structure of name servers. More than 30 years later, this same system is still very much in use, making Paul Mockapetris the official Father of DNS. Figure 1-1 shows a timeline that summarizes the evolution of DNS through 1987.

Brief History of DNS

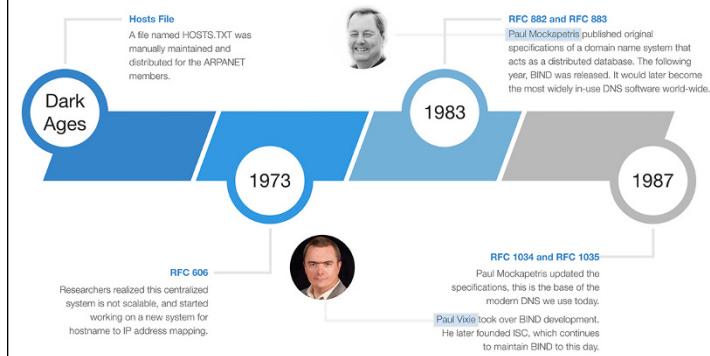


FIGURE 1-1: It took only ten years to get from unconnected computers to the modern DNS we know today.

DNS Structure

DNS distributes responsibility for an ever-growing list of network device names. It does this by creating a hierarchy of responsibility. This is often shown with an upside down tree, such as Figure 1-2, where the root servers are at the top and the leaves (which represent all the end host nodes on the Internet) are at the bottom. The entire tree represents the namespace of DNS. Each server that is responsible for part of the namespace is called a “name server.” Some name servers just send packets along until they reach an answer.

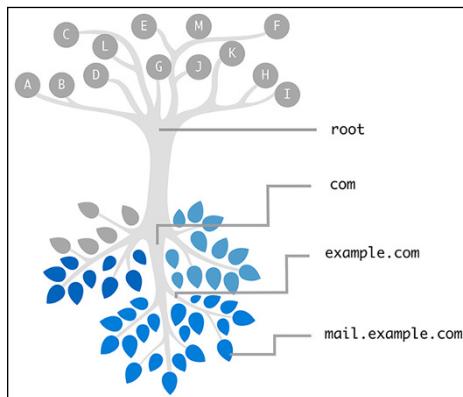


FIGURE 1-2: Like the branches of a tree, each domain name can have multiple subdomains.

The *root name servers* direct DNS queries to name servers for each of the top-level domains, which are the main branches just below it (for example, .com, .net.jp, and .info). *Root name servers* are authoritative name servers for DNS's root zone, which is sometimes written as a single dot (.). Being *authoritative* for a zone means being responsible for that domain, except the parts delegated to different authoritative name servers.



TECHNICAL STUFF

Name resolution is the process of following these delegations of responsibility until reaching the name server that has the answer: in other words, the authoritative server for that zone. Root name servers are a critical part of the Internet infrastructure because they represent the first step in name resolution; thus every name server in the world needs to know about them in order to walk down the tree to the end host it is looking for. The list of root name servers, including their names and addresses, resides on every DNS server in a file known as the *root hints file*.

This allows a company like "Example," shown in Figure 1-2, to register the domain name "example.com" and manage just the subdomain names within that domain. The rest of the world doesn't need to know where Example's name servers are. When an Internet user wants to visit example.com, the user's device can ask the root name servers, which will send it to .com name servers, which will, in turn, send it to the example.com name servers. The example.com name servers have answers for any subdomain names within example.com.

Authority and Zones

A DNS zone is a domain that a party is responsible for maintaining, minus any subdomains the party delegated control of to another party. The responsible party uses that zone to maintain the resource records for that domain. *Resource records* map information to common names. The server where the party edits the resource records is typically called the *primary name server* or *master name server*. Because a single server isn't enough for a robust solution, additional name servers can also be authoritative for a zone by getting a copy of the zone data from the primary or master name server through a process called a *zone transfer*. These additional servers are called *secondary name servers* or *slave name servers*.

The terms *master* and *primary* are interchangeable. The data on the primary server is the only version that a person should ever edit. Similarly, the terms *secondary* and *slave* (an unfortunate old term) are also interchangeable. The secondary server receives information only as a copy of the data on the primary server. Nobody should ever edit the data directly on the secondary server.

Resource Records

Resource records identify the information and/or services associated with a given domain name.

All resource records use the same format, which we discuss in the following list:

- » **Name:** A domain name in which this resource record pertains.
- » **TTL:** A 32-bit integer that specifies the time interval that the resource record may be cached before it should be discarded.
- » **Class:** Two octets specifying the class of the data in the RDATA field. The most common type is IN for Internet.
- » **Type:** This field specifies the meaning of the data in the RDATA field.
- » **RDLENGTH:** A 16-bit integer that specifies the length in octets of the RDATA text: for instance, how large is the payload.
- » **RDATA:** A variable-length string that describes the resource. The format varies according to the TYPE and CLASS of the resource record.

Although all resource records share a common overall structure, they may contain different types of information in their RDATA field, such as network- or service-specific information.



REMEMBER

To understand resource records, you need to understand the NAME field a bit more. The NAME field contains a domain name associating this name to various information. If the information is about the domain itself that is enough, but when the information is related to an end host then a fully qualified domain name or FQDN is used. The FQDN has two parts: the host name and the domain name.

For example, consider FQDN `www.example.com`. In this FQDN, `www` is the host name, and `example.com` is the domain name. Each word that is separated by the dot character is also known as a *label*, so `www` is a label, `example` is a label, and `com` is a label.

There can be more than one resource record associated with an FQDN. All the resource records associated with an FQDN that have the same values in the NAME and TYPE fields, regardless of their RDATA value, are considered a *Resource Record Set (RRSET)*.

Common Resource Records and Their Uses

Each of the following sections discusses a type of record and lists the most important fields in the record with example data. However, we have omitted two important fields from each section:

- » We omitted the CLASS field because, for all common record types, the value is IN for Internet.
- » We omitted the RDLENGTH field because it is just a reference to the length of the data. It could be any value.

A records

A records are the most common record used in DNS. These records match easily remembered hostnames to the IP addresses of the resource. Figure 1-3 is a sample A record.

NAME	TTL	TYPE	RDATA
<code>eat.example.com</code>	3600	A	93.184.216.34

FIGURE 1-3: The RDATA field in an A record contains the IP address of the resource.

AAAA Records

AAAA records (or *quad A*, as they are often called) are used to map hostnames to their IPv6 addresses. Figure 1-4 is a sample quad A record.

NAME	TTL	TYPE	RDATA
eat.example.com	3600	AAAA	fc00::dead::beef

FIGURE 1-4: The RDATA field in a quad A record contains the IPv6 of the resource.

SOA records

The SOA record, which stands for *start of authority* records, provides the querier, including secondary servers and recursive servers, information about the zone itself, including the master name server (*mname*), the responsible party (*rname*), and timers for how the zone and its records should be handled.

The timers that appear in the RDATA field are:

- » **Refresh:** How often a secondary server should contact the primary server for updates.
- » **Retry:** How soon a secondary server should try contacting primary server if an attempt fails.
- » **Expire:** How long a secondary server can hold the zone data when it cannot reach the primary server.
- » **Minimum:** How long recursive name servers can cache a negative answer, such as NXDOMAIN, also known as *NCACHE* for negative cache.

Figures 1-5 and 1-6 show a sample SOA record.

NAME	TTL	TYPE	RDATA
example.com	3600	SOA	sns.dns.icann.org. noc.dns.icann.org. 201610710 7200 3600 1209600 3600

FIGURE 1-5: An SOA lookup returns a lot of information in the RDATA field compared to other resource records, including the *mname* and *rname*.

MNAME	RNAME
example.com.	sns.dns.icann.org. noc.dns.icann.org. (
3600 IN SOA	
201610710 ; serial	
7200 ; refresh (2 hours)	
3600 ; retry (1 hour)	
1209600 ; expire (2 weeks)	
3600 ; minimum (1 hour)	

FIGURE 1-6: This figure shows much of the same information from the RDATA field in the previous figure, as it appears in the dig tool.

Other Record types

Other record types include the following:

- » **CNAME** records map additional names back to a hostname, like an alias. A common use of this kind of record is to map service names to a server name. For example, the record `ftp.example.com` points to the server named `helium12.example.com`.
- » **PTR** records map an IP address back to the hostnames that exist for the IP address. PTR records are mostly used by applications and systems to determine the name associated with the IP address.
- » **NS** records publish the name of the name servers for a domain. This is used by other DNS servers to find the authoritative name servers of a domain.
- » **MX** records allow DNS clients to find mail servers for the domain.
- » **TXT** records allow an FQDN to be associated with a text string. TXT records originally provided information about the host associated with that name. Today they are more often used in security by helping to verify the host by providing hashes and other information.

Query Path, Recursion, and Iteration

A *query path* is the set of queries starting from the initial question from the client and finishing with the answer the client receives. A query path can be as simple as a client asking a server and receiving an answer directly. However, a query path can also be complex and include multiple servers working together to track down the answer. Understanding the query path of a given question allows you to troubleshoot issues and identify where you need to focus your DNS security.

It's important to understand what we mean when we say a *client* asks a question. When a DNS client asks simple questions like "What is the IPv4 address of `www.google.com`?" a *stub resolver* is the piece of software code that actually sends the DNS question. For the scope of this book, you can assume that "client" is an

application or a machine that has a stub resolver running on it. Therefore, a web browser is a client, and a laptop or mobile phone can also be a client.

When a client asks a question, it is unable to follow referrals given by other name servers to track down the answer on its own. It has to rely on a full-fledged DNS server, which may call on other name servers, to chase down the answer. Figure 1-7 illustrates several name servers involved in answering a simple question.

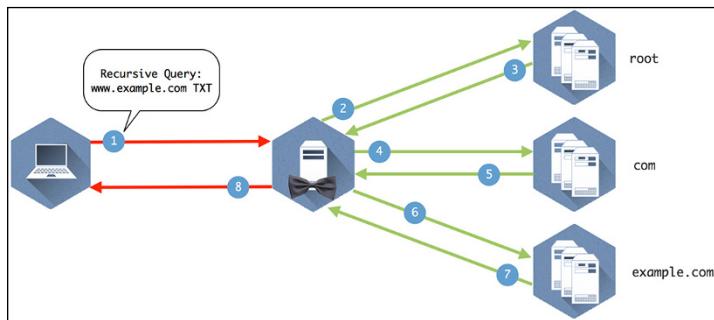


FIGURE 1-7: Eight queries and answers make up the query path required to get this client the answer to “What is the IP address of `www.example.com`?”

Now take a look at the different parts of the query path in Figure 1-7 and break down how the servers use *recursion* and *iteration* to move down the DNS tree and find the answer the client is looking for.

Recursion is the process repeatedly asking the question to name servers and following referrals until finding the name server with the answer. The recursive query basically says, “I would like to know the answer to this question. And if you don’t know the answer, please ask others until you’ve found the answer.”

All clients ask recursive queries by default, because clients are usually not capable of “walking the tree” to chase down the answers on their own. A name server providing recursion accepts recursive queries, and fulfills them by executing iterative queries in the background to track down answers. As a result of processing recursive queries, recursive name servers build up a rich cache of answers over time, thus they are also known as *caching name servers*.

An *iterative name query* is typically sent by DNS servers to other DNS servers, in pursuit of finding the answer. A key difference

between iteration and recursion is that iterative queriers must have the ability to follow referrals, which means they track down the answer.

Figure 1-8 provides the details to the query path to illustrate how it all comes together.

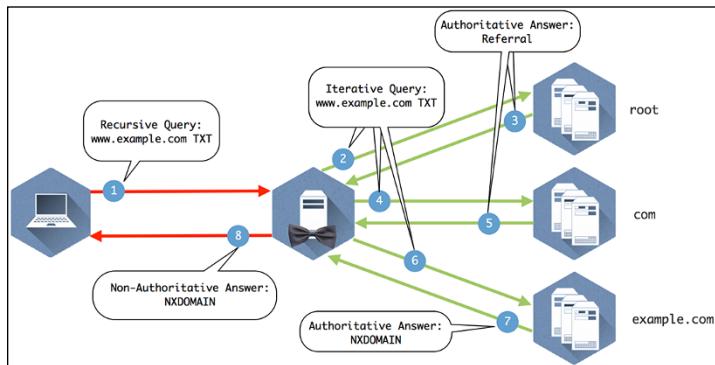


FIGURE 1-8: The query path.

Here's a breakdown of how the query path happens:

1. The client queries the recursive name server that it is configured to use.
2. The recursive server checks its internal cache. If it doesn't find the answer to the question, it checks its root hints file and sends the query to one of the 13 root name servers listed in the file.
3. The root name server doesn't contain specific records that answer the question, but it does know where the com name servers are. It sends a referral in the form of the NS RRSET for the com name servers and the A records for those name servers. These matching A records are called *glue records*.
4. The recursive server caches the responses from the root name server and queries one of the com servers it was given for www.example.com.

5. The com name servers do not contain www.example.com, but someone has registered example.com and provided com with the information for the name servers. The com name server sends a referral of the NS and A records for example.com.
6. The recursive server caches the records from com and queries one of the example.com name servers provided in the referral.
7. The example.com name server returns the answer from the authoritative zone example.com.
8. The recursive name server caches the answer returned from the example.com name servers and sends the response to the client.

IN THIS CHAPTER

- » Understanding how DDoS attacks work
- » Learning about cache poisoning attacks
- » Discovering how malware uses DNS

Chapter 2

Threats to DNS Security

DNS is becoming a more common target of network attacks. As one of the oldest and most relied-on protocols of the modern Internet, DNS is the cornerstone to almost all other services and protocols. This makes DNS an appealing target to attackers.

Because it is one of the most relied-on protocols, stopping attacks can't be as simple as adding a firewall rule. It's good to know how these attacks work before discussing solutions to stop them. In this chapter, we show you a few common types of DNS-based attacks; in the next chapter, we focus on remediation and solutions.

DDoS Attacks

There are many types of *distributed denial of service* (DDoS) attacks. You probably see them in the news these days, often accompanied by a ransom to induce the attacker to stop the DDoS attack. When it comes to DNS, you can look at specific types of attacks that are used to overwhelm DNS servers, thus rendering the DNS service unavailable. When an attack on the DNS is successful, it can bring an organization to a screeching halt. When a company can't publish the addresses for its web and mail servers, business stops.

The two main attack methodologies you want to look at are *amplification* and *reflection*. While technically two different attack tactics, attackers often combine amplification and reflection attacks.

Amplification

An *amplification attack* is a technique where a small query can trigger a large response, such as querying for a TXT record or a zone transfer when you haven't secured zone transfers to only your trusted sources. By flooding the server with short requests that require long responses, even a relatively weak computer can overload a DNS server. The DNS server is so busy doing the heavy lifting to respond to all these bogus requests that it doesn't have time to respond to legitimate ones.

DNS servers make surprisingly good amplifiers. Here is a simple example:

If a user makes a DNS query for “`isc.org/ANY`,” the query is 44 bytes long, and the response is 4077 bytes long. That is around 93 times amplification!



WARNING

This example and some simple math show how devastating amplification can be. Say an attacker is generating queries with a *botnet* (a network of independent hosts, or bots, infected with malware), and each bot has a measly 1 Mbps connection to the Internet:

With a 1 Mbps connection, each bot could send the 44 byte query from the previous example approximately 2,909 times per second.

Because each query results in a response that is 4,077 bytes, you can calculate that the DNS server has to come up with about 93 megabytes each second.

If the botnet contains 11 bots all doing the same thing, that's a total of over 1 gigabytes that the DNS server is supposed to send out every second of the attack!

Amplification alone makes an effective attack, because attackers can use less powerful resources to overload more powerful servers. However, what makes DNS an even greater target for devious DDoS is reflection.

Reflection

A *reflection attack* sends queries that look like they came from the victim of the attack. The response (often a large, amplified answer) is sent to the victim, who never asked, and the amount of the response traffic could potentially overwhelm the victim's network.

In a reflection attack, an attacker sends a query to a recursive name server with a spoofed source IP address. Instead of his real IP address, he places the target (victim) IP address as the source IP address. The recursive name server does the legwork, retrieves the answer to the query from the authoritative name server, and sends the answer to the unsuspecting victim.

Combination attacks

Now the attacker combines the two techniques by spoofing the victim's IP address and sending a carefully crafted query that will result in a large payload. This is a very effective DDoS attack; the authoritative name server provides the amplification, and the recursive name server provides the reflection. This allows the attacker to attack two different victims at the same time. It also causes the victim of the amplification attack to possibly believe he or she was attacked by the second victim, causing potentially even more mayhem.

Figure 2-1 shows an example of combining amplification and reflection to cause DDoS. As you can see, this attack consumes the resources of not only the victim, but also those of the recursive name servers. This is why securing a publicly facing recursive server against recursion from the world is one of the most important best practices in DNS Security. It prevents you from playing the role of amplifier.

Here is how the attack proceeds:

1. An attacker-controlled botnet sends DNS queries with a victim's spoofed IP address to recursive name servers.
2. Recursive name servers follow the standard DNS path and send the query to other authoritative name servers.
3. Recursive name servers receive the (amplified) response from authoritative servers.
4. Recursive name servers send the (amplified) response to the victim, whose IP address was listed as the source address in the original query.

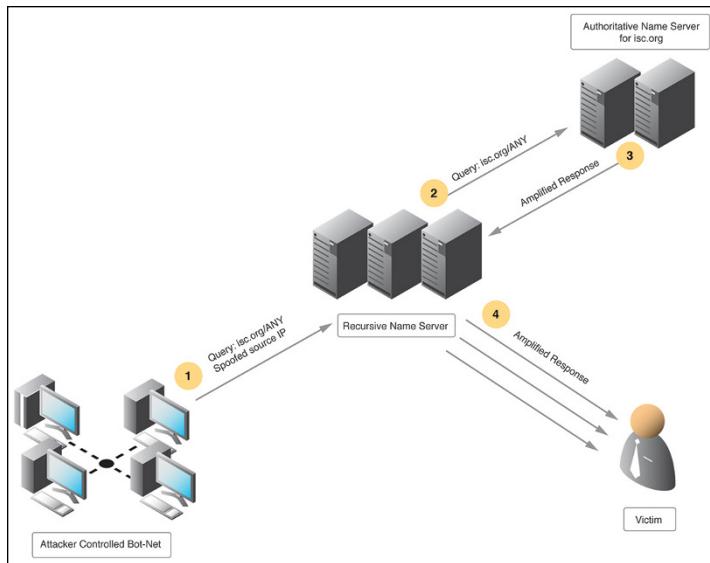


FIGURE 2-1: An amplified reflection attack.

Cache Poisoning

Also known as *DNS spoofing*, cache poisoning focuses on corrupting the cached answers on the recursive name servers, either through software exploits or protocol weaknesses.

Figure 2-2 describes an older-style DNS cache poisoning by attaching malicious answers to the ADDITIONAL section of the DNS response. An older, less secure recursive name server would accept the RRset in the ANSWER section, and also accept the RRset in the ADDITIONAL section, even if those records were unrelated to the question being answered.

Here's how that variant of the cache poisoning attack works:

1. Query a recursive server for foo.example.com.
2. Send replies spoofing the example.com name server's IP address and with various query IDs, including malicious records in the Additional Section, in this case redirecting all queries to a com name server to the bad guy's DNS server at 10.17.34.25. (Of course, this would need to be a reachable IP address in a real attack.)

DNS Cache Poisoning

DNS cache poisoning corrupts a DNS server's cache with bogus data such as a rogue address, opening the door to data theft of logins, passwords, and user credit card numbers, as well as other threats.

1. Attacker queries a recursive name server (of a service provider or enterprise) for a subdomain that doesn't exist (e.g. q0001.ABCcorp.com)
2. The recursive name server does not have the IP address and queries an ABCcorp.com name server
3. Before the ABCcorp.com name server can send NXDOMAIN response, the attacker sends lots of spoofed responses that look like they are coming from a legitimate ABCcorp.com server.

Spoofed responses map www.ABCcorp.com to the IP address of a server controlled by the attacker

4. The recursive name server accepts a spoofed response and caches the record
5. A user queries the recursive name server for the IP address of www.ABCcorp.com
6. The recursive name server replies to the user with a cached rogue IP address
7. The user connects to a site controlled by attacker, which may look exactly like the real ABCcorp website

How The Attack Works

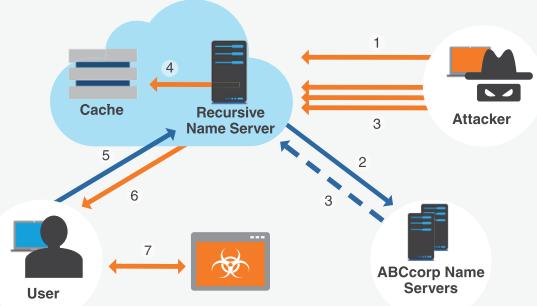


FIGURE 2-2: A cache poisoning attack can land unsuspecting website visitor on nefarious pages.

```
; ANSWER SECTION:  
foo.example.com 3600 IN A 10.17.34.25  
;; ADDITIONAL SECTION:  
a.gtld-servers.net. 1540000 IN A  
10.17.34.27 ; (bad guy's IP address)
```



REMEMBER

Name servers have been patched against this older cache poisoning attack for some time. This was accomplished by requiring DNS servers not to accept additional data that is unrelated to the original query.

One of the most (in)famous cache poisoning attacks was discovered by an American computer security expert, Dan Kaminsky,

in 2008. The attack Kaminsky discovered takes advantage of a weakness in the DNS protocol itself. When a recursive name server makes a query, it sets the value of five fields. If it receives a response with matching values for these fields, with an answer to the question that it asked, then it accepts the answer as legitimate, returns it to the client, and stores it in its cache.

As it turns out, four of the fields are easy to figure out and spoof, so only one field is actually meaningful: the Query ID or TXID. The Query ID has relatively few possible values, by computer standards. It is a 16-bit number, or a value between 0 and about 64,000. This makes it possible for an attacker to initiate an information request, then when the recursive server asks the next server up the tree, the attacker tries his luck firing a lot of responses at the recursive name server that look like they came from that authoritative server. If the attacker gets lucky and guesses the right query ID, the original server stores the bad data from the attacker's bogus response in cache and passes it on if any other servers ask for it.

A well-crafted Kaminsky attack can insert a bogus entry in the cache within minutes. To make matters worse, the attacker could choose to poison an NS record, thus taking command of an entire domain rather than just a single entry. The most common fix for the vulnerability sends queries from random source ports, since the response must arrive at the same random port. Having two values to guess correctly makes crafting an acceptable bogus response much more difficult. The attack becomes more time-consuming, but there is no surefire way to guard against this type of attack without changing the protocol itself. Fortunately, the enhanced DNSSEC helps with this problem. We talk about this in Chapter 3.

Malware and Exfiltration

The term *malware* comes from combining the words “malicious” and “software.” Malware is a category of software designed with malicious intent. Often, malware is installed without the user’s knowledge and operates in the background, safely hidden from attention.



REMEMBER

Although malware is typically distributed via web servers, disguised as part of the web content, its foundation is in DNS. Modern attackers don't rely on a single, registered domain name such as `malware.not-evil.example.com`. Instead, they create a complex system that automatically registers hundreds or thousands of domain names on the fly and configures them to point to web servers that distribute malicious content. Changing their domain names constantly makes it difficult to block them at a DNS level.

Once it has infected a computer, malware uses DNS to resolve a list of domain names of possible command-and-control servers until it finds one that is available and responds. Command-and-control servers are special servers maintained by the malware owners that tell the infected client what to do. The list of command-and-control servers can be compiled into the malware, or synthesized using what's known as a domain-generation algorithm.

Malware can also sneak sensitive data out of your network, a technique known as *data exfiltration*. Because it is done over DNS, data exfiltration is also known as *DNS tunneling*.

Although there are many DNS tunneling implementations out there, all of them rely on the capability of clients to perform DNS queries. DNS tunneling software allows users to do:

- » Relatively innocuous things, such as getting free airport Wi-Fi
- » Potentially dangerous acts, such as using SSH to evade corporate firewalls
- » Outright malicious activities, such as stealing sensitive information

In the case of the last, clients evade detection by breaking data down into query-sized chunks, disguising sensitive data as DNS queries, and sending them to malicious DNS servers on the far end who can unpack these queries and reconstruct the data.

It's like stealing someone's car without opening the garage door: You have to break the car down into small chunks that fit through the doors and windows, and then rebuild the car outside. Except in the case of data exfiltration, the malware breaks down files, sometimes even encrypting each chunk, before sneaking them off your premises to reassemble.



Figure 2-3 illustrates a simplified flow of how data exfiltration works over DNS. The steps in Figure 2-3 go like this:

1. The attacker registers the domain name ZG5ZC2VJDXJPDHKK.COM, and sets up name server NS1.ZG5ZC2VJDXJPDHKK.COM.
2. The infected client encodes stolen information, in this case the text “Pa\$\$w0rd” into “UGEKJHCWCMQK.”
3. The client makes the DNS query for the domain with the encoded password as a subdomain: UGEKJHCWCMQK.ZG5ZC2VJDXJPDHKK.COM.
4. A recursive name server finds the authoritative name server NS1.ZG5ZC2VJDXJPDHKK.COM and sends the query there.
5. The attacker recognizes the subdomain value as the encoded password. The attacker decodes the information UGEKJHCWCMQK back to recover “Pa\$\$w0rd.”

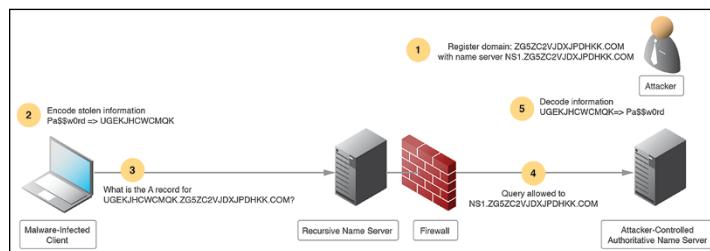


FIGURE 2-3: DNS tunneling can sneak your passwords into DNS queries that look like gibberish.

In this example, it is not necessary for the client to receive a response from the malicious server, because the goal was to send information out; however, the process can easily include the malicious server sending back an exploit to be executed on the infected client.

An example of a well-publicized malware attack that leveraged DNS is WannaCry. WannaCry was (or is) *ransomware* — a type of malware that encrypts valuable data on a computer and charges the user a price to get it unencrypted — that used a vulnerability in the Windows platform at the time. Once inside the network, it could infect any unpatched Windows computer using SMB, Microsoft’s remote file sharing protocol. Users of the infected computers saw a screen like in Figure 2-4.



FIGURE 2-4: Unlike attacks that just cause mayhem, ransomware attacks your pocketbook directly.

Where DNS was noticeably involved was in WannaCry’s decision to encrypt data. The malware incorporated a kill-switch using DNS, first checking for the existence of an obscure domain name via a DNS query and, when it was not found, encrypting files and holding the computer hostage. If the creators of the malware needed to stop the ransoming they could just register the domain.

Once the “good guys” discovered this fact, they could address WannaCry in two different ways.

1. DNS providers could look for queries for that domain name to identify clients infected with WannaCry.
2. Good guys could register the kill-switch domain name that told the malware not to ransom data, in effect neutering the ransomware until the code could be changed to point to other not-yet-registered domain names.

In this example, the malware developer’s use of DNS led to the possibility of easy remediation. In the next chapter, we look at more ways to prevent and resolve DNS attacks.

IN THIS CHAPTER

- » Using response rate limiting to prevent amplification
- » Creating a trustworthy Internet through DNS security extensions
- » Configuring response policy zones

Chapter 3

DNS Security Solutions

The good guys are not idle in the face of all these threats, of course. The security community responds quickly to new kinds of attacks by creating security mechanisms and protections to keep your systems safe. Even before that, *white hat* hackers, or ethical computer security experts, try to penetrate their own systems so they can uncover weaknesses before attackers have a chance to exploit them.

In this chapter, we explore some of the more important security mechanisms that have been added to DNS since its original design. These mechanisms can help mitigate the threats we discuss in Chapter 2. We also look specifically at how Infoblox has made implementing these protections user-friendly.

Response Rate Limiting (RRL)

A DDoS attack overwhelms DNS servers with so many superfluous queries that they can't correctly direct legitimate traffic. When you look at addressing DDoS attacks, the critical issue becomes separating valid queries from malicious queries. Attackers are very good at making their queries look legitimate, so the trick is to look at the *pattern* of queries rather than at the queries themselves. This is the fundamental concept behind *response rate limiting* (RRL).



TECHNICAL STUFF

RRL basics

If a DDoS attack is like a single doughnut-store customer buying up all the doughnuts so that no one else can have any, then RRL is limiting each customer to only one doughnut of each flavor each day. If each person can buy only one chocolate and one glazed each day, then the DDoS attack can't do as much damage.

RRL depends on the concept of a *token bucket* to limit how many responses the DNS server will send to any single client. A configuration on the DNS server creates a token bucket for each client that contains, for example, five tokens and adds five more each second. The server takes out one token for each query response it sends to the client. Figure 3-1 illustrates the setup.

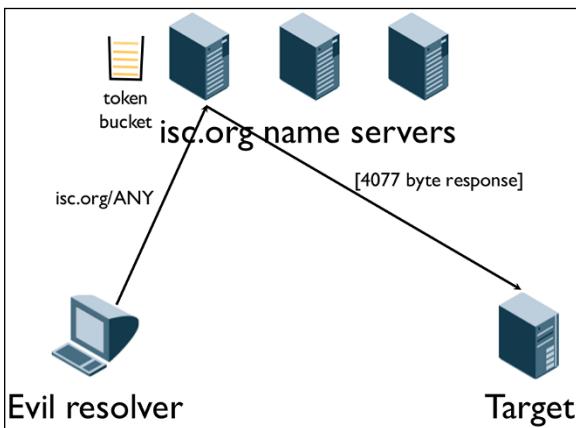


FIGURE 3-1: The token bucket prevents name servers from sending responses to the same client too frequently.

Once the bucket is empty, it takes 15 seconds to refill. In the meantime, the server sends truncated responses to the client instead of full responses. Truncated responses prevent the amplification that attackers get from full query responses. Add in the 15-second delay preventing the volume of queries from overwhelming the server, and presto! Suddenly the DNSSEC server is not useful in amplification attacks.

When legitimate clients happen to empty their own token bucket, waiting 15 seconds to try again isn't too much of an imposition. Alternately, the client can retry the query right away over TCP.

Infoblox Advanced DNS Protection (ADP)

Advanced DNS Protection (ADP) is how Infoblox allows users to navigate complex RRL configuration through a simplified graphical interface and apply rules targeted to specific attacks. Since different attacks have specific DNS traffic patterns, ADP can let you choose the attacks you want to focus on.



TIP

Targeting your RRL to specific attacks allows more legitimate queries to get through and punishes the bad queriers more by limiting their access to resources even further.

Additionally, Infoblox lets you push and modify these changes across all the authoritative DNS Infoblox appliances at one time. Figure 3-2 shows ADP working to block attacks.

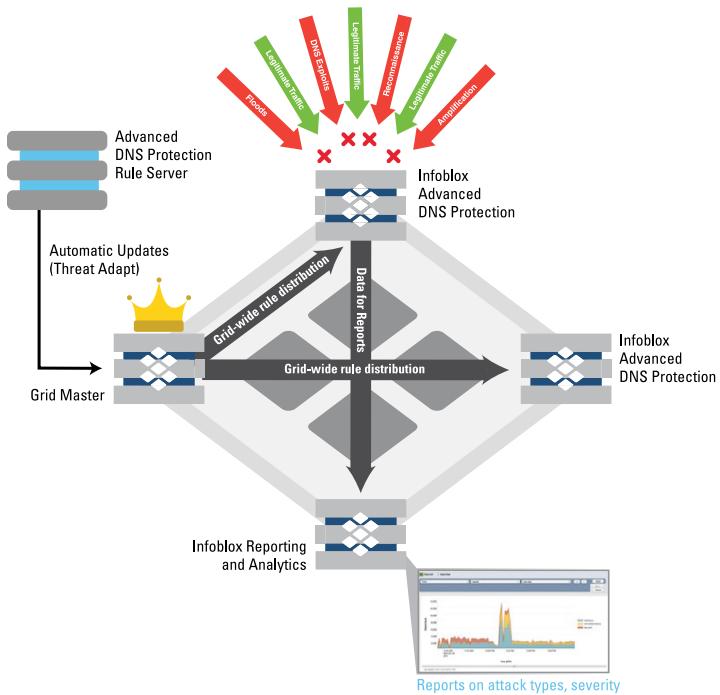


FIGURE 3-2: Infoblox Advanced DNS Protection simplifies configuring DNSSEC.

Pairing ADP with an Infoblox Reporting and Analytics Server adds visibility to your security (see Figure 3-3). Being able to see attacks and then adjust on the fly can be as important as stopping them in many cases — especially against modern attackers willing to change their attack profiles as soon as they meet resistance.



FIGURE 3-3: Reporting and analytics servers offer a window into attempted attacks as they happen.

DNS Security Extensions (DNSSEC)

The lack of security in DNS has been a well-known issue for decades, one that the DNS community has been trying to solve since the late 1990s. These efforts resulted in the birth of the *DNS Security Extensions* (DNSSEC).

DNSSEC basics

Take a look at what DNSSEC does and doesn't provide:

- » **DNSSEC provides answer validation.** It does this through the magic of public key cryptography. Public key cryptographers use a pair of keys, one being public and one being

private, that allows a DNS server to validate that the answer to a query came from the sender it says it did (in other words, that the sender wasn't spoofed) and that the message content wasn't changed during transmission. Keys are used to "sign" zones, meaning to create a cryptographic representation of the data that can only be turned back with the other key. This provides both message authentication and message integrity.

- » **DNSSEC is backwards compatible.** DNSSEC can be deployed side-by-side with traditional DNS. If a domain is not yet DNSSEC-enabled, a DNSSEC-aware name server falls back to using traditional DNS.
- » **DNSSEC doesn't encrypt traffic.** Although DNSSEC uses both public and private keys, they're only used for authentication purposes. Someone eavesdropping on the wire can still see all the DNS messages in plaintext; it just becomes nearly impossible to spoof.

DNSSEC requires deployment on both recursive name servers and authoritative name servers: The recursive name servers ask for additional security information and perform validation checks, while authoritative name servers provide signed resource records in responses. Figure 3-4 shows a simplified explanation of a DNSSEC-aware recursive name server, also known as a validating resolver, checking the answer it received.

In each step, the validating resolver asks for additional security information, chasing up to the domain's parent for more security information if necessary, until it has reached the root server. The root server's public key is well known, and is the only key that is trusted by the validating resolver in most cases.

Cache poisoning (discussed in Chapter 2) is a very difficult attack to guard against, and its damage can be tremendous. When properly deployed, DNSSEC stops cache poisoning attacks because attackers can no longer impersonate authoritative name servers.

In addition to stopping cache poisoning and other DNS-based attacks, DNSSEC also bolsters other security features. Many organizations already publish anti-spam information such as SPF (Sender Policy Framework) or DKIM (DomainKeys Identified Mail) in DNS.

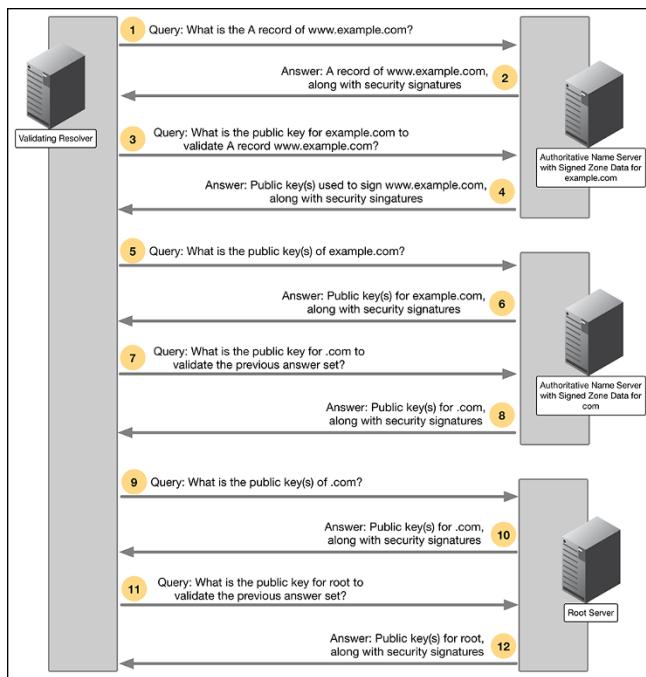


FIGURE 3-4: An example conversation between DNSSEC servers.

There are other security-related applications of DNS, as well. For example, you can publish the SSH fingerprint of servers in DNS, so when a user logs in to a machine via SSH, she can verify she's not being tricked to log in to a different machine. You can also publish TSLA records, which contain information about your SSL certificates, and when someone visits your website, he can verify that his HTTPS connection has not been hijacked, and the certificate he received over HTTPS matches the one you published in DNS.



REMEMBER

Being able to authenticate DNS messages is a big step forward not just for DNS, but also for the Internet as a whole, because this would be the first time we have a global database that is trustworthy.

Infoblox DNSSEC

Infoblox provides both DNSSEC validation and zone signing. One-click signing and validation make this the simplest solution for managing DNSSEC-signed zones.

Figure 3-5 shows the interface for configuring the DNSSEC settings for an Infoblox Grid.

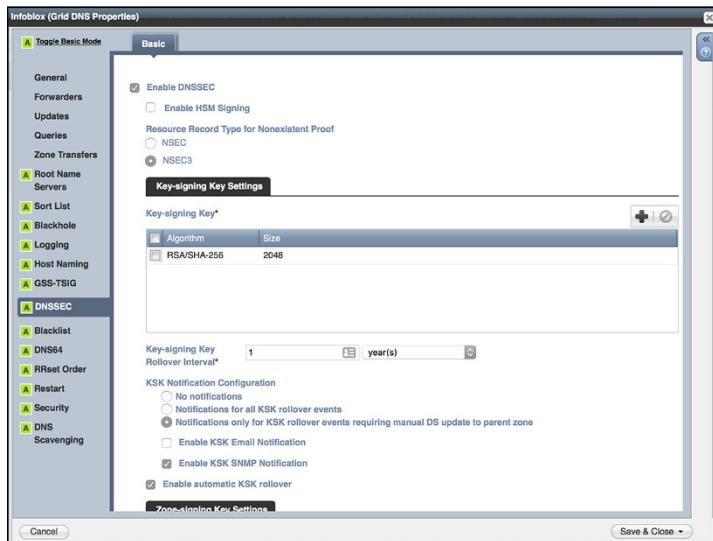


FIGURE 3-5: Scrolled-down view of the configuration screen.

After the global settings and DNSSEC are enabled, you can sign zones with just one click. Figure 3-6 shows how to start signing zones.

Infoblox also automates DNSSEC maintenance tasks, such as key rollovers (importing and exporting keys), and sends email alerts to remind administrators of important actions and due dates. Figure 3-7 shows this.

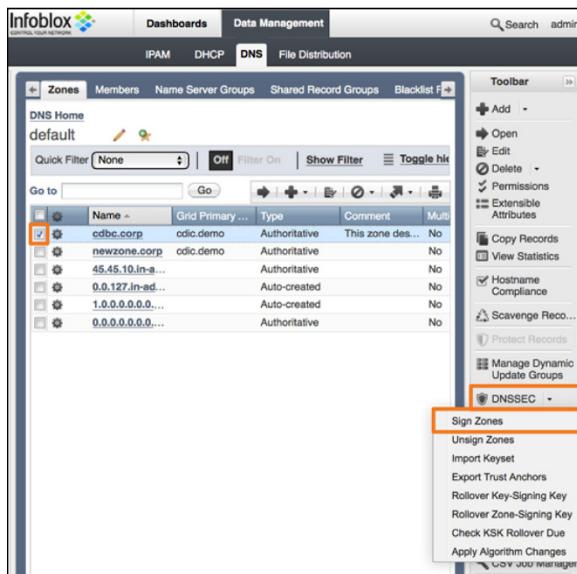


FIGURE 3-6: Signing zones in Infoblox.

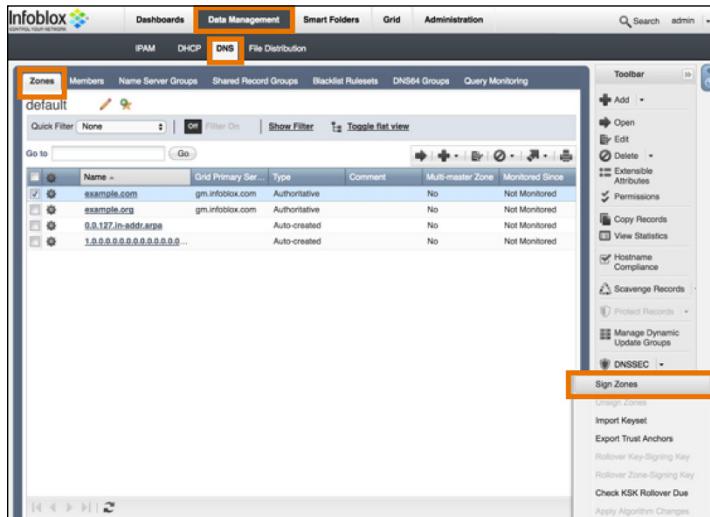


FIGURE 3-7: Automating maintenance tasks.

Response Policy Zones

Response policy zones (RPZs) are a way for you to control what your queriers can and can't look up using a recursive DNS server. By understanding the reputation of the servers and services that clients are querying, you can determine actions to take when the recursive server receives queries for certain domain names or sees information in the DNS responses that point to those malicious servers.

RPZ basics

The overall idea of how RPZ works is that you can create policies for how to handle specific queries (or responses) and choose which of a number of possible actions to take — like redirecting the client to an internal security page — and store those policies in special authoritative zones on your DNS servers. You can also share these zones by transferring them from DNS server to DNS server.

In order to create these RPZs, you need to understand the concept of *reputation*. Reputation describes a zone's history of offering malicious content. Several reputation services track and analyze the providers of malicious content. These services have the (for most) unenviable task of chasing the bad guys, predicting their next move, and even determining the algorithms they use to generate new bad domains. Thankfully, these services also publish this reputation data for everyone else to use.



TECHNICAL STUFF

RPZs make policy data available in DNS zones. The policy data is then transferred between servers using conventional DNS protocols.

The resource records in the zone are expressions of DNS policy, which apply to domain names with the NAME field (QNAMEs) or to the target data (RDATA). Note that the RDATA field in DNS consists of one or more subfields carrying the actual payload for a resource record.

The owner name of a RPZ QNAME policy resource record set (RRSET) is the relative name of the domain triggering the filter. In the case where the policy zone is provided by Infoblox and

called `rpz.infoblox.com`, the policy affecting responses to `nasty.comelook.badsite.net` would be attached to the domain name `nasty.comelook.badsite.net.rpz.infoblox.com`. In this example, `nasty.comelook.badsite.net` is the site impacted. If all the FQDNs in `badsite.net` were bad, the wildcard `*.badsite.net.rpz.infoblox.com` would catch all subdomains under `badsite.net`.

The event that determines whether a query or response matches a specific entry is called a *trigger*. We describe the possible triggers in RPZ in the following sections.

QNAME trigger

The QNAME trigger operates on the NAME field of a query. A wildcard format can block a site and all subdomains — for example, `*.badguys.com`. The QNAME trigger should follow the full query and response cycle, including iteration, if required. The recursive server must follow the delegation hierarchy related to processing name server (NS) resource records and corresponding A and AAAA records.

IP trigger

The IP trigger matches on the IPv4 or IPv6 address in the RDATA field of resource records in a DNS response. This is useful when you know the IP address is bad, no matter what its name might be. For example, if the known bad IP address is 10.11.12.13, and the client queried for example.com MX record, if the answer contains 10.11.12.13, then the answer is blocked from the client.



WARNING

IP trigger can be a double-edged sword. While it is very effective at blocking a known bad IP address, it will also block any answer that contains that IP address. In the age of cloud computing, many websites may share the same IP address. If one of the sites causes the IP address to be blacklisted, it could end up getting every other site on the same IP address blocked.

Client IP trigger

The client IP trigger matches on the source IP address of the client initiating a query. Since it matches a specific client IP address, it is very targeted and is very effective when used to block hosts that are known to be compromised. It can send all queries from a single client to a remediation server, for example.

NSDNAME trigger

The NSDNAME trigger matches the name of the authoritative name server found during recursion. This is an extremely powerful trigger targeting an entire DNS name server. This trigger affects all the domains served by the name server. Since the NSDNAME trigger works during recursion, this trigger must be used with extreme caution and only when an entire name server is known to be malicious.

NSIP trigger

The NSIP trigger matches the IP addresses in IPv4 A records and IPv6 AAAA records corresponding to name server records associated with domains (glue records). It checks these against NSDNAME policy records, which protects against name servers with multiple or changing names. The NSIP trigger can block a name server and all the domain names it serves.

RPZ responses

Once a trigger matches a record in a response policy zone, you can take one of the four following policy actions:

- » **NXDOMAIN:** An RRset consisting of a single CNAME whose target is the root domain “.” causing a response of NXDOMAIN (“There is no such domain name.”).
- » **NODATA:** The RRset consists of a single CNAME whose target is the wildcard top-level domain (*.), which results in a response of NODATA (“There is no data of that type attached to that domain name.”).
- » **NO-OP/PASSTHRU:** This RRset consists of a single CNAME whose target is the query name itself and expresses an exception preventing any policy action for the name. You can think of this type of response as a “pass-through,” permitting domain information to pass through a filter unencumbered. Because a rule matched in the zone, you will see an entry in the logs, providing information to the administrator.
- » **Local data:** An RRset matching the QTYPE of the original query. In the event an RRset exists but does not match the QTYPE of the query, the name server sends a NODATA response.

If this section is of specific interest to you, much of the material is from an Infoblox training course on RPZ. You can learn more at <https://www.infoblox.com/support/infoblox-education-services/> or email startlearning@infoblox.com.

Infoblox DNS firewall

Infoblox's DNS firewall brings you RPZ configuration and more. The Infoblox Grid Architecture is particularly powerful when it comes to RPZs, allowing you manage and control all your authoritative servers through one simple interface. Step-by-step wizards simplify the complexity of maintaining rules for local RPZ zones. See Figure 3-8.

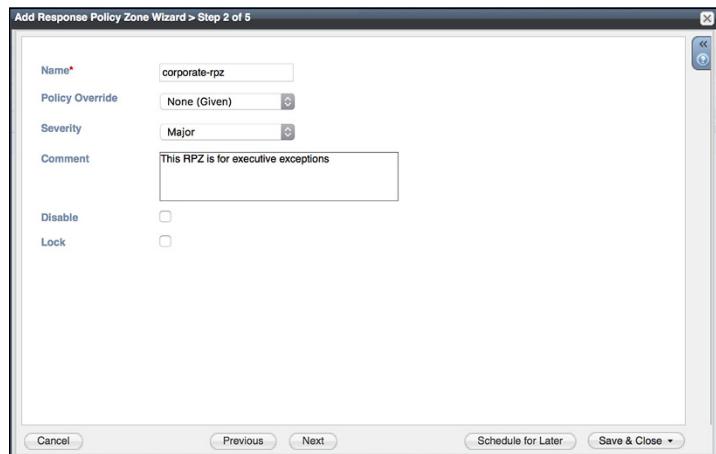


FIGURE 3-8: Infoblox provides simple wizards that allow you to create your own Response Policy Zones.

Infoblox provides reputation data with its Threat Intelligence Feed products. While there are numerous feeds available, they break down into four main categories:

- » **Base feed:** Protects against known malicious threats that are dangerous as destinations.

- » **Malware feed:** Protects against known malicious threats that can take control of your system.
- » **Ransomware feed:** Protects against attacks that encrypt your data until you pay the attacker to decrypt it.
- » **Bogon feed:** Protects against *bogons*, bogus IP addresses, which are commonly the source addresses of DDoS attacks. Many ISPs and end-user firewalls filter and block bogons, because they have no legitimate use, and usually are the result of accidental or malicious misconfiguration.

Again, simplified wizards in Infoblox walk you through adding these feeds to your members, as you see in Figure 3-9.



FIGURE 3-9: Infoblox provides simplified wizards to add your RPZ feed information for all servers in your Infoblox Grid.

Additionally, Infoblox offers cloud-based RPZ feed management through its Infoblox Security Portal. Here you can select the feeds that best suit your needs and also assign which Grid members can access them. See Figures 3-10 and 3-11.

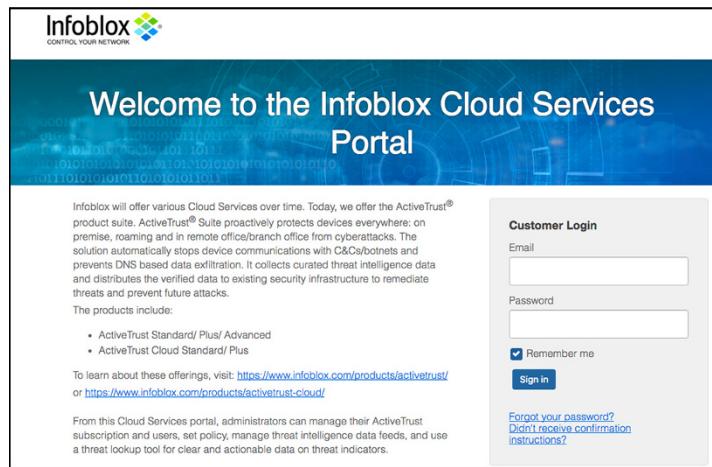


FIGURE 3-10: Infoblox Security Portal allows you to self-administer your RPZ Feed data through the Infoblox secure web portal.

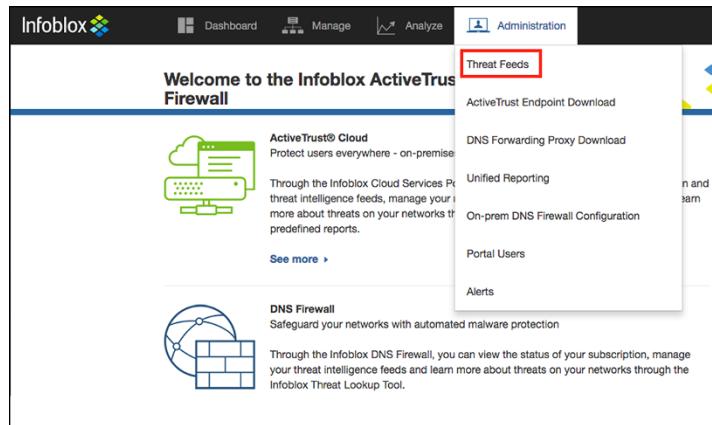


FIGURE 3-11: The Infoblox Security allows you to manage all aspects of the feeds your Grid requires.

IN THIS CHAPTER

- » Hardening DNS infrastructure
- » Ensuring resiliency
- » Blocking communication with known malicious sites
- » Leveraging response policy zone feeds
- » Sharing indicators of compromise (IoCs)

Chapter 4

Ten Keys to Improving DNS Security

Now that you know the bases, this chapter discusses ten actions you should put high on your priority list to protect yourself.

Use Dedicated DNS Appliances

If you host your own DNS servers, make sure to use the right hardware. Deploying DNS on general-purpose servers can be risky because:

- » The default configurations are not set for DNS. Ports associated with other services are open by default, and therefore vulnerable to attack.
- » Users on general-purpose servers often have operating-system-level account privileges on the server, including administrator rights. Users with administrator rights can create vulnerability, sometimes even accidentally.

» You have to apply time-consuming updates manually, and that means downtime for maintenance windows. People have a bad habit of putting off downtime, which is even worse, because the servers are vulnerable to the attacks that the updates would have protected against.



TIP

Instead, you should employ a dedicated DNS hardware appliance or non-open-source DNS software.

Keep DNS Server Software Up-to-Date

As with any other computer application, service, or protocol, new DNS vulnerabilities continuously crop up. Attackers dedicate a lot of time to discovering these weaknesses and figuring out how to exploit them.



REMEMBER

That's why keeping your DNS server software updated with the current software versions and security updates is a job that you can never permanently cross off your to-do list. Whether you find a dedicated appliance that applies updates for you or have to apply updates manually, you simply must stay on top of it.

Have an Onsite DNS Backup

Even if you outsource your DNS to a managed DNS service provider, you should host your own dedicated backup DNS server.

Neither Internet service providers nor managed DNS service providers are impervious to attack. In 2016, DNS service provider DYN and Internet service provider Deutsche Telekom were both victims of massive DDoS attacks that caused widespread outages.

A coordinated attack on your vendor isn't the only reason to have a backup. More commonly, hardware or network failures can cause slow DNS performance or an outage.

Avoid Single Points of Failure

A *single point of failure* is a part of your network that, if it stops working, shuts down the entire process. Eliminating single points of failure throughout any system or network is a basic principle of secure, resilient design.



TIP

One important way to avoid single points of failure is to have multiple Internet links from different ISPs pointing to your websites. By introducing different ISPs, you increase the authoritative DNS servers that cache your links and reduce the risk of cache poisoning diverting your visitors.

Run Authoritative DNS Servers inside DMZs

If attackers manage to compromise an authoritative DNS server, they can change the DNS data of any domain for which that server is authoritative. The effect can be devastating. These changes quickly replicate across the Internet and, in some cases, take days to fix.

Stop these problems before they start by setting up your authoritative DNS servers *inside* a secure network demilitarized zone (DMZ). The DMZ allows the importing of DNS records only from a secure primary server that is also located inside your DMZ.

Turn Off Recursion

As much as possible, you want to control who can ask your authoritative DNS server for information. You can restrict zone transfers to the specific IP addresses of your secondary DNS servers, for example, to prevent attackers from getting hostnames and IP addresses for your network. For another example, you can digitally sign your zone transfer records to prove their authenticity.

Use Threat Intelligence

Threat intelligence is information about your network's weakest points and the most likely attacks you are likely to receive. You can use this information to make decisions and set priorities about how to protect your company.

Use Response Policy Zones

A Response Policy Zone (RPZ) allows you to set policies for specific domains. For example, when your DNS server receives a request for an A record for a certain domain, you can create a policy to do one of the following actions instead:

- » Return a nonexistent domain (NXDOMAIN) error.
- » Return no data at all.
- » Redirect the domain so that you can further observe and analyze any malicious activity associated with the domain.

Use IPAM

As your network grows, even keeping visibility into everything becomes a challenge. With an enterprise-grade IP Address Management (IPAM) solution, you can consolidate information about your core network infrastructure into one comprehensive and authoritative database. This solution lets you see your entire network topology, including:

- » Network assets
- » IP addresses and switch ports
- » Virtual local area networks (VLANs)
- » Usernames

Automate Security Tasks whenever Possible



TIP

Tasks that you can automate with DNS security software include many common scenarios:

- » When your DNS security solution detects DNS-based data exfiltration or malware from an infected host, it should notify an endpoint security solution to clean the infected endpoint.
- » When a new device joins the network, your DNS security solution should trigger a vulnerability scan.
- » Until the completion of the vulnerability scan and mediation of any problems, your DNS security solution should trigger a network access control (NAC) solution to prevent the endpoint from getting on the network until it is compliant.



IS SOMETHING MISSING FROM YOUR SECURITY STRATEGY

You've put a lot into antivirus, firewalls and endpoint solutions to keep the bad guys out. But here's the thing. They're going right around your security because your DNS is wide open.

The Domain Name System (DNS) is the #1 attack vector for malware and a commonly used pathway for ransomware, hijacking, and data theft.

With Infoblox, you can close the gaps in DNS that traditional security measures leave unprotected.

Our solutions stop the broadest array of DNS-based threats in their tracks—automatically, completely, and in real time.



SECURITY. IT'S IN OUR DNS™

Learn more at: www.infoblox.com/threat-center/

Harness context-aware security to combat threats

Distributed denial-of-service (DDoS) attacks, malware and ransomware, and data exfiltration from enterprise networks all have an increasingly common thread — attackers use your DNS infrastructure against you. Whether it's to cause a massive service outage, redirect network traffic to malicious sites, communicate with command-and-control (C&C) infrastructure, or surreptitiously send valuable data from your network, DNS has become an essential part of cybercriminals' playbook.

Inside...

- Understand the history and fundamentals of DNS
- Understand how DNS is critical to any security solution
- Protect critical infrastructure from attackers leveraging DNS
- Block malware propagation
- Prevent data exfiltration over DNS



Joshua M. Kuo is the Director of Services at DeepDive Networking. **Robert Nagy** is the Founder and CTO at DeepDive Networking. **Cricket Liu** is the Chief DNS Architect and a Senior Fellow at Infoblox

Go to **Dummies.com**[®]
for videos, step-by-step photos,
how-to articles, or to shop!

for
dummies[®]

A Wiley Brand

ISBN: 978-1-119-43731-4
Not for resale

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley's ebook EULA.