CS 4404 - Mission 2

Matthew Hagan, Peter Maida, Tim Winters

November 2019

1 Introduction

Domain Name Service, or DNS for short is a method of naming resources that are connected to the internet. The goal of it is to connect various domain names to specific services behind them. In layman's terms, DNS is like a contact app on a mobile phone. When you try to call somebody, most often you don't know their phone number (anymore); rather, you look up their contact and call them that way. DNS is very similar, as it allows the user to use an easy to remember name to connect to intended service.

Keeping with the analogy, if you were to change someone's phone number, say for the contact 'Dad', to another like the number for 'Papa Johns', then when the client calls what they assume to be 'Dad', it will actually call 'Papa Johns'. Now imagine the publisher of yellow-pages had a vested interest in wanting you to contact a certain auto repair company instead of another. All it would need to do would be to list the wrong phone number in the phonebook. Or if you called the phone operator and asked for the number of the same auto repair company, the phone operator may relay the wrong number, and give you the phone companies preferred repair company.

We can perform these same attacks using DNS, by interferring with the client-server connection in two ways. Overwriting the DNS record, and altering a DNS response packet.

1.1 Bombast

ISPs in general are very powerful resources. They provide users with the data in question. If they did not care about laws and regulations, they could easily sell your traffic or serve you a service that is not consented.

Bombast's ISP is no different. Given the amount of power the ISP has, they can easily misdirect a user without them even knowing. Yes, a trained user (maybe a Cyber Security professional) may be fishy of this, but the average user may not. In their eyes, getting served the wrong service could just be seen as a "computer glitch". With this in mind, it is important to ensure that DNS performs as expected, as it was born during a different internet, one more trusting in design.

Our goal in this research mission is to first build a infrastructure that allows the Bombast ISP to redirect the user away from its competitor. After that, a back and forth attack and defense will be designed and implemented to demonstrate various ways to prevent the ISP from meddling and the corresponding countermeasures that could be deployed to help embody "the CEO's vision".

2 Reconnaissance

2.1 DNS Security Goals

Based on how DNS currently works, it does not actively meet all of the four security goals. According to the National Institute of Standards (NIST), "The primary security goals for DNS are data integrity and source authentication" [1].

2.1.1 Authenticity

The authenticity of a DNS domain is important as the goal of DNS as a service is to point a user from a

domain to the intended service behind that domain. If the domain is not who it says it is, the user will be misdirected and inevitably not reach the service that they were trying to reach.

2.1.2 Availability

Availability of a DNS domain is very important. As DNS domains are often targeted for DOS or DDOS attacks, it is important to keep them online as accessing the resource in question can only be done if the user knows the domain name or the IP address, which is far less common [1]. To note, if a DNS domain is unavailable, it does not mean that the service itself is unavailable. Rather, it means that the service in question will be a lot harder to access as the domain which users rely on would be not behaving as expected.

2.1.3 Confidentiality

Confidentiality is not a security goal of DNS [1]. This is primarily because DNS is designed to be public. As such, keeping the DNS data private would defeat it as a service, as nobody would be able to access it.

2.1.4 Integrity

While it might not be transparent at first, data integrity is also important to DNS as a service. As accessing the service that a given domain points to involves information in transit, it is crucial that that information is not tampered with as it is moving, as that would be an easy way for a attacker to quickly misdirect a user to a different or unintended service that will inevitably result in the user experiencing unintended side effects.

2.2 ISP Vantage Point

Bombast has two unique vantage points because all traffic between you and the internet must go through them, and because often times they provide DNS servers to their customers. Traffic includes communication with the DNS server. If this communication is not secure, either through encryption to guarantee integrity, or signing to guarantee authenticity, then the ISP could modify the contents of the packet for their own gain.

The ISP can also decide to modify DNS records directly, preventing clients from using secure tunnels to ensure integrity.

This mission will focus on exploiting the ISPs ability to spoof records and snoop and modify DNS responses that don't align with their goals.

2.3 Ramifications of DNS Misuse

DNS is an important service used to translate between host names and IP addresses; however, the implementation of the protocol leaves it vulnerable to exploits. Since DNS is critical for Internet operations, countless operating systems, and applications, it needs to be protected in order to not be used maliciously [2].

If the DNS service does get breached, an adversary is able to deny the usage of the DNS protocol with various Denial of Service (DoS) attacks. Adversaries are able to poison the DNS cache with fake Resource Records to redirect users to different sites, overwhelm a user's resources with responses to fake requests, or overwhelm the resources on the DNS server [2].

2.4 DNS Amplification and Reflection Attacks

A DNS amplification and reflection attack is a Distributed Denial of Service (DDoS) attack that works by generating a huge number of responses from DNS open resolvers. An attacker sends a message to the DNS open resolver that will respond with a large response, an amplification of the previous request. The message also includes a forged IP address that will send the large response from the DNS open resolver to the client under attack, thus reflecting the response to a different target. A victim becomes overloaded with the large responses and are denied from using the DNS service. This can be strengthened from a Denial of Service attack to a Distributed Denial of Service attack by using multiple DNS open resolvers so that

the effect on the target device is magnified [2] [3].

A very simple, but effective way to defend against amplification and reflection attacks is to rate limit your traffic. This will prevent an adversary from consuming all of your resources during an attack. However, this solution still does not solve the larger issue and restricts you from using the internet to the fullest.

3 Infrastructure

The infrastructure consists of five different virtual machines (VMs) with the following roles:

1. Customer Client
2. Bombast ISP
3. DNS Server
4. Bombast Web Server
5. Competitor Web Server

The connection between the various VM nodes can be seen in Figure 1. The client wants to go to google.com, so it sends its query to the DNS server which travels through the ISP. The DNS replies with the IP address for google.com to the client which travels through the ISP. The client then connects to the google.com web server to get the desired content.

3.1 Bind Install

For our DNS server, we chose to setup a authoritative NS. The reason for this is that no resources taken away by other tasks. Beyond that, it allows for performance replies To install bind as a authoritative DNS, the first steps are as follows:

```
$ sudo apt-get install bind9 bind9utils
    ↪ bind9-doc

Need to Nano/vim:
    -/etc/named.conf

Then search for:
    -"recursion yes;"
and change to:
    -"recursion no;"
```

At this point, you are left with a authoritative NS server.

Because the ISP acts as a gateway to the DNS server, it is able to monitor all traffic that flows between *Client* and *DNS*. In this attack, the bombast ISP is able to intercept all packets sent by DNS to Client, and modify the records to conform to Bombast's business model. In this scenario, Bombast's goal is to route all requests to *spectrum.com* to *bombast.com* by overwriting the DNS record response.

4 Attack and Defense

4.1 Description

ISPs have a unique set of powers with regards to DNS zones. Because most ISPs provide their customers with a router, and the router is initially configured to use the ISPs DNS server, the ISP has two vantage points to interfere with the client's DNS queries.

We will explore an attack and a defense for both of these vantage points. The first attack will be the ISP directly changing the Spectrum IP in the DNS server configuration files, and the second attack will be the ISP meddling with DNS records coming from a third party DNS server. The defense against the first attack will be changing the DNS server the client uses to not use the ISP-owned server, and the response to the second attack will be enabling DNSSEC so that traffic is hidden from the ISP.

4.2 Performing the Attack

4.2.1 Intial State

Before we begin attacking the DNS queries, let us establish what the different queries should result in under normal conditions.

Without ISP meddling, the response of nslookup requests to spectrum.com and bombast.com are

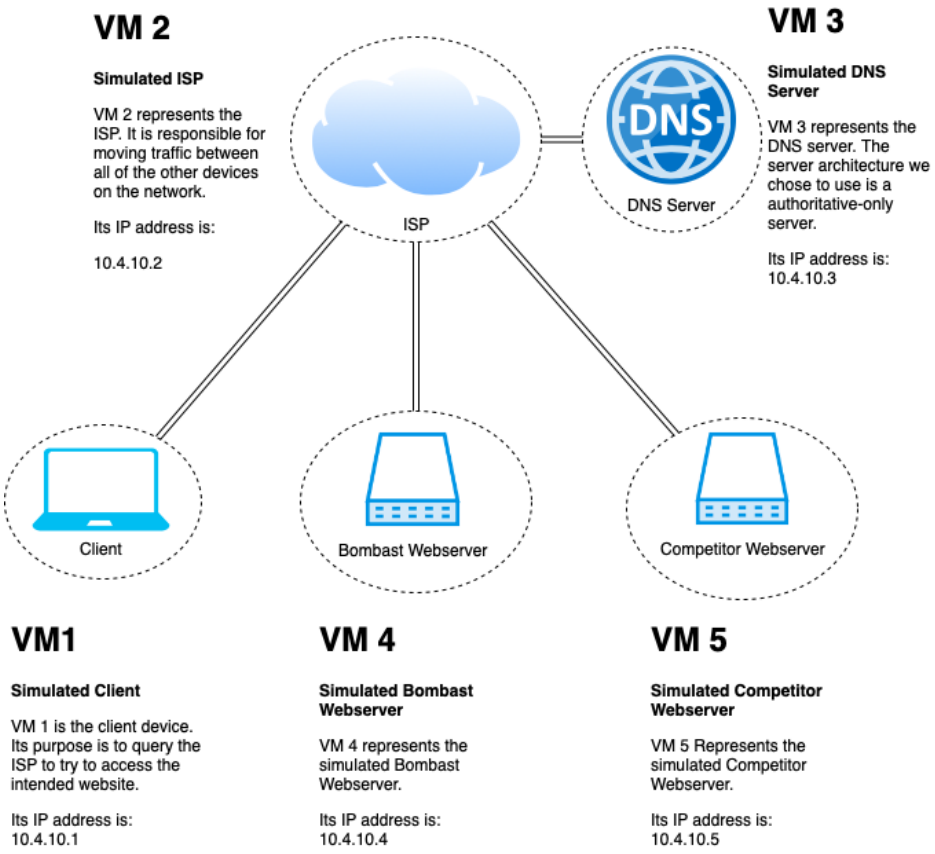

Figure 1: The model showcasing the route of the DNS query.

```
> nslookup spectrum.com
Server:  10.4.10.129
Address: 10.4.10.129#53

Name: spectrum.com
Address: 10.4.10.5

> nslookup bombast.com
Server: 10.4.10.129
Address: 10.4.10.129#53

Name: bombast.com
Address: 10.4.10.4
```

4.2.2 Attack Phase 1

The first step of this attack simulates the ISP manually changing the DNS records for spectrum.com. If the ISP controls the DNS server, they can simply modify the zone files for bind.

The normal configuration for the spectrum zone file is shown in the following figure

```
root@cs4404:/etc/bind# cat db.spectrum.com
;
; BIND data file for local loopback interface
;
$TTL 604800
$INCLUDE "keys/Kspectrum.com.+008+02640.key" #zsk
$INCLUDE "keys/Kspectrum.com.+008+26797.key" #ksk
@       IN      SOA     spectrum.com. root.spectrum.com. (
                              2         ; Serial
                         604800               ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )     ; Negative Cache TTL
;
@       IN      NS      spectrum.com.
@       IN      A       10.4.10.5
```

By changing the IP address of the A record for spectrum.com in `/etc/bind/db.spectrum.com.`, Bombast can redirect all traffic meant for `10.4.10.5` to `10.4.10.4`

After this attack, the new nslookup response shows the IP address for bombast.com.

```
> nslookup spectrum.com
Server:  10.4.10.129
Address: 10.4.10.129#53

Name: spectrum.com
Address: 10.4.10.4

> nslookup bombast.com
Server: 10.4.10.129
Address: 10.4.10.129#53

Name: bombast.com
Address: 10.4.10.4
```

Note how spectrum and bombast point to `10.4.10.4`

4.3 Defense Phase 1

The defenses for this sort of attack are limited. The defense we implemented was to circumvent the attackers attempt to limit access to spectrum.com by switching DNS providers to a 'third-party' DNS (we call this AltDNS), that is, one that is not controllable by Bombast, and therefore has no desire to alter DNS records to support them.

To simulate the AltDNS, we simply 'revoke' Bombast's access to the DNS zone settings by return the zone settings to its original settings i.e. changing the A record ip back to `10.4.10.5`

4.4 Attack Phase 2

Bombast is not satisfied. They do not want their customers circumventing their unethical business practices. To enforce the supervisor's policy Bombast has decided to modify the DNS response from AltDNS.

To alter the records, the ISP runs a middle-man packet filter using the Python packages netfilterqueue and scapy to intercept packets and modify their contents. If the IP address of the DNS response matches that of a known Spectrum IP address, the filter will change the IP address to point to bombast.com. The sniffed traffic can be seen in Apendix H with DNSSEC implemented.

The result of this attack are similar to Attack Phase 1 to the client where the user will still be redirected to bombast.com.

4.5 Defense Phase 2

To counteract this attack, we must look to use DNSSEC to hide the response from the ISP meddler. By enabling DNSSEC on the bind9 service running the AltDNS, we can have secure communication between the client and DNS without Bombast being able to determine if we are querying spectrum.com.

To enable and configure bind, we mainly used the article from DigitalOcean on "How To Setup DNSSEC on an Authoritative BIND DNS Server". We begin by modifying the `/etc/bind/named.conf.options` file and adding a few attributes to enable DNSSEC as referenced in Appendix E. Next we generate both our Zone Signing Key and Key Signing Key with keygen as referenced in Appendix B. Afterwards, we modify the zone file to include the new keys as referenced in Appendix C. Once this is complete, we are able to use the `dnssec-signzone` command to sign the zone file. We will update our local configuration file to include this new signed file as referenced in Appendix D. Next we reload BIND and check to see if the files got properly signed with `dig spectrum.com +dnssec +multiline` as referenced in Appendix F and Appendix G. Finally, we verify that the client received the correct results with the same `dig spectrum.com +dnssec +multiline` command on the client end as referenced in Appendix I. This output is the same as on the server side and will have a status of NOERROR when the data is correctly verified.

4.6 Defending Against Amplification and Reflection

In order to defend against a DNS amplification and reflection attack that Bombast may consider implementing in the future, add rate limiting so that the client will not be completely DoSed from the network. We have implemented this by using a library in Python called ratelimit to limit the amount of calls an API can have in a given time frame. Twitter recommends that time frame be 15 minutes.

5 Conclusion

This paper, as the title suggests, shows that DNS as it stands contains a large amount of security problems. Beyond that, it shows that Internet Service providers have a lot of power in the absence of various security measures that can be taken, such as DNSSEC.

To demonstrate this, we first attacked the ISP directly by changing the Spectrum IP in the DNS server configuration files. To defend against this, we changed the DNS server the client uses to not be the one that the ISP owns. After this, to demonstrate further problems, we as the ISP meddled with the DNS records coming from a third party DNS server. To defend against this, we as a team implemented DNSSEC to allow the traffic to be hidden from the ISP.

In the scope of this paper, implementing DNSSEC was our largest problem that we encountered. This was due to a multitude of reasons, from following various guides that performed 'clever' shortcuts that ultimately made it more difficult to extrapolate what they were doing to cumbersome configuration issues, such as signing, validation, and zone issues.

Overall, this Mission provided a welcome challenge. It pushed our knowledge and patience to the limits. If we were to do it again, it would have been exciting to look more into VPNs in the scope of securing DNS. This was something that we as a team did extensively look at, but ultimately decided against implementing in the final submission due to the notable infrastructure overhead that comes with setting up a OpenVPN tunnel. VPN as a service would have allowed us to make sniffing through tools like Wireshark/Tshark useless as it creates a end to end tunnel for the DNS query to be sent over.

References

[1] R. Chandramouli and S. Rose, "Secure domain name system (dns) deployment guide," *NIST*, no. Special Publication 800-81-2. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-81-2.pdf

[2] Cisco, "Dns best practices, network protections, and attack identification." [Online]. Available: https://tools.cisco.com/security/center/resources/dns_best_practices

[3] A. S. Jose and A. Binu, "Automatic detection and rectification of dns reflection amplification attacks with hadoop mapreduce and chukwa," pp. 195–198, 2014, iD: ieee$_s$6906023.

A Server Dig Before DNSSEC

```
cs4404@cs4404:~$ sudo /etc/init.d/bind9 reload
[sudo] password for cs4404:
[....] Reloading bind9 configuration (via systemctl): bind9.service.
cs4404@cs4404:~$ dig @localhost spectrum.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @localhost spectrum.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13108
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;spectrum.com.                  IN      A

;; ANSWER SECTION:
spectrum.com.           604800  IN      A       10.4.10.5

;; AUTHORITY SECTION:
spectrum.com.           604800  IN      NS      spectrum.com.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Nov 20 20:15:21 EST 2019
;; MSG SIZE  rcvd: 71
```

Figure 2: Output from "dig spectrum.com +dnssec +multiline" on the DNS server end.

B Key Generation

Figure 3: The creation of both ZSK and KSK

C Updated Zone File with Keys

```
[root@cs4404:/etc/bind# cat db.spectrum.com
;
; BIND data file for local loopback interface
;
$TTL 604800
$INCLUDE "keys/Kspectrum.com.+008+02640.key" #zsk
$INCLUDE "keys/Kspectrum.com.+008+26797.key" #ksk
@       IN      SOA     spectrum.com. root.spectrum.com. (
                              2         ; Serial
                         604800               ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )     ; Negative Cache TTL
;
@       IN      NS      spectrum.com.
@       IN      A       10.4.10.5
```

Figure 4: The new zone file after keys are manually included.

D Conf Local

```
[root@cs4404:/etc/bind# cat named.conf.local
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
zone "bombast.com" {
        type master;
        file "/etc/bind/db.bombast.com";
};
zone "spectrum.com" {
        type master;
        file "/etc/bind/db.spectrum.com.signed";
};
```

Figure 5: The local configurations include the newly signed file.

E Conf Options

```
root@cs4404:/etc/bind# cat named.conf.options
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================
        dnssec-validation auto;
        dnssec-enable yes;
        dnssec-lookaside auto;
        key-directory "/etc/bind/keys/";

        auth-nxdomain no;    # conform to RFC1035
        listen-on-v6 { any; };
};
```

Figure 6: The option configuration file includes the key directory as well as the dnssec enabled option.

F Server Dig Before DNSSEC Signing

```
cs4404@cs4404:~$ sudo /etc/init.d/bind9 reload
[sudo] password for cs4404:
[....] Reloading bind9 configuration (via systemctl): bind9.service.
cs4404@cs4404:~$ dig @localhost spectrum.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @localhost spectrum.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13108
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;spectrum.com.                  IN      A

;; ANSWER SECTION:
spectrum.com.           604800  IN      A       10.4.10.5

;; AUTHORITY SECTION:
spectrum.com.           604800  IN      NS      spectrum.com.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Nov 20 20:15:21 EST 2019
;; MSG SIZE  rcvd: 71
```

Figure 7: Output from "dig spectrum.com +dnssec +multiline" on the DNS server end before the files get signed with dnssec-signzone

G Server Dig After DNSSEC

```
[root@cs4404:/etc/bind# dig spectrum.com +dnssec +multiline

; <<>> DiG 9.10.3-P4-Ubuntu <<>> spectrum.com +dnssec +multiline
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62283
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;spectrum.com.          IN A

;; ANSWER SECTION:
spectrum.com.           604800 IN A 10.4.10.5
spectrum.com.           604800 IN RRSIG A 8 2 604800 (
                               20191221010224 20191121010224 2640 spectrum.com.
                               tiizT4kD24lL+DHPOURnOj5hLfQ80iCllH94EaiZIK7I
                               QmPWTgHR6BFZNMM1damtfjols6fuiuSgV88uWMCTy7L7
                               IF+ql8ZEVhSecTucMa4GfTtnJgm103RbraMeCy8FFq0C
                               9NoL41b7IIBEe8/Bn7o1QGJ7/p0rMfkmurapY2M= )

;; AUTHORITY SECTION:
spectrum.com.           604800 IN NS spectrum.com.
spectrum.com.           604800 IN RRSIG NS 8 2 604800 (
                               20191221010224 20191121010224 2640 spectrum.com.
                               xIMrIK5zKx8ZZ9wej1hrmoIw84uXsnjE+ghUPAfx4Dni
                               DTD6M7iavD0srHcKwJCCrRHZ5GvHw69TnvzNnFYTEUlR
                               sqYQf1XSm8/fNOTpi7aFvpl9s/p018IZh0nz9ju1BGIg
                               YsdRcMEqyrYijeUEtxzoVacKjndwV5/pTLhU7dU= )

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Nov 20 21:31:17 EST 2019
;; MSG SIZE  rcvd: 415
```

Figure 8: Output from "dig spectrum.com +dnssec +multiline" on the DNS server end.

H MITM Interception Traffic

```
[cs4404@cs4404:~$ sudo python dns_intercept.py
Before:
###[ DNS Resource Record ]###
  rrname    = 'spectrum.com.'
  type      = A
  rclass    = IN
  ttl       = 604800
  rdlen     = None
  rdata     = 10.4.10.5
###[ DNS RRSIG Resource Record ]###
  rrname    = 'spectrum.com.'
  type      = RRSIG
  rclass    = IN
  ttl       = 604800
  rdlen     = None
  typecovered= A
  algorithm = RSA/SHA-256
  labels    = 2
  originalttl= 604800
  expiration= Sat, 21 Dec 2019 01:02:24 +0000 (1576890144)
  inception = Thu, 21 Nov 2019 01:02:24 +0000 (1574298144)
  keytag    = 2640
  signersname= 'spectrum.com.'
  signature = '\xb6(\xb3O\x89\x03\xdb\x89K\xf81\xcf9Dg:>a-\xf4<\xd2 \xa5\x94\x7fx\x11\xa8\x99 \xae\xc8Bc\xd6N\x01\xd1\xe8\x11Y4\xc35u\xa9\xad~:%\xb3\xa7\xee\x8a\xe4\xa0W\xcf.X\xc0\x
93\xcb\xb2\xfb _\xaa\x97\xc6DV\x14\x9eq;\x9c1\xae\x06};g&\t\xb5\xd3t[\xad\xa3\x1e\x0b/\x05\x16\xad\x02\xf4\xda\x0b\xe3V\xfb \x80D{\xcf\xc1\x9f\xba5@b{\xfe\x9d+1\xf9&\xba\xb6\xa9cc'

None
After:
###[ DNS Resource Record ]###
  rrname    = 'spectrum.com.'
  type      = A
  rclass    = IN
  ttl       = 604800
  rdlen     = None
  rdata     = 10.4.10.4
###[ DNS RRSIG Resource Record ]###
  rrname    = 'spectrum.com.'
  type      = RRSIG
  rclass    = IN
  ttl       = 604800
  rdlen     = None
  typecovered= A
  algorithm = RSA/SHA-256
  labels    = 2
  originalttl= 604800
  expiration= Sat, 21 Dec 2019 01:02:24 +0000 (1576890144)
  inception = Thu, 21 Nov 2019 01:02:24 +0000 (1574298144)
  keytag    = 2640
  signersname= 'spectrum.com.'
  signature = '\xb6(\xb3O\x89\x03\xdb\x89K\xf81\xcf9Dg:>a-\xf4<\xd2 \xa5\x94\x7fx\x11\xa8\x99 \xae\xc8Bc\xd6N\x01\xd1\xe8\x11Y4\xc35u\xa9\xad~:%\xb3\xa7\xee\x8a\xe4\xa0W\xcf.X\xc0\x
93\xcb\xb2\xfb _\xaa\x97\xc6DV\x14\x9eq;\x9c1\xae\x06};g&\t\xb5\xd3t[\xad\xa3\x1e\x0b/\x05\x16\xad\x02\xf4\xda\x0b\xe3V\xfb \x80D{\xcf\xc1\x9f\xba5@b{\xfe\x9d+1\xf9&\xba\xb6\xa9cc'
```

Figure 9: The sniffing that the man in the middle is able to perform. Note how they modified the ip address.

I Client Dig After DNSSEC

```
[root@cs4404:~# dig spectrum.com +dnssec +multi

; <<>> DiG 9.10.3-P4-Ubuntu <<>> spectrum.com +dnssec +multi
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24479
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;spectrum.com.          IN A

;; ANSWER SECTION:
spectrum.com.           604800 IN A 10.4.10.4
spectrum.com.           604800 IN RRSIG A 8 2 604800 (
                               20191221010224 20191121010224 2640 spectrum.com.
                               tiizT4kD24lL+DHPOURnOj5hLfQ80iCllH94EaiZIK7I
                               QmPWTgHR6BFZNMM1damtfjols6fuiuSgV88uWMCTy7L7
                               IF+ql8ZEVhSecTucMa4GfTtnJgm103RbraMeCy8FFq0C
                               9NoL41b7IIBEe8/Bn7o1QGJ7/p0rMfkmurapY2M= )

;; AUTHORITY SECTION:
spectrum.com.           604800 IN NS spectrum.com.
spectrum.com.           604800 IN RRSIG NS 8 2 604800 (
                               20191221010224 20191121010224 2640 spectrum.com.
                               xIMrIK5zKx8ZZ9wej1hrmoIw84uXsnjE+ghUPAfx4Dni
                               DTD6M7iavD0srHcKwJCCrRHZ5GvHw69TnvzNnFYTEUlR
                               sqYQf1XSm8/fNOTpi7aFvpl9s/p018IZh0nz9ju1BGIg
                               YsdRcMEqyrYijeUEtxzoVacKjndwV5/pTLhU7dU= )

;; Query time: 24 msec
;; SERVER: 10.4.10.129#53(10.4.10.129)
;; WHEN: Wed Nov 20 22:20:28 EST 2019
;; MSG SIZE  rcvd: 475
```

Figure 10: Output from "dig spectrum.com +dnssec +multiline" on the client end.