Implementing DNSSEC under the .pr ccTLD

Luis Alberto Medina Ramos
Computer Engineering
Jeffrey Duffany, Ph.D
Department of Electrical & Computer Engineering and Computer Science
Polytechnic University of Puerto Rico

Abstract *— When the Internet was developed it was not designed with security in consideration. It currently relies on the Domain Name System protocol for name resolution which is home to many types of exploits. The Internet Engineering Task Force developed a set of extensions to address these security concerns that resulted in the Domain Name System Security Extensions protocol DNSSEC. Users of this protocol will benefit from authentication and data integrity in a medium which has been historically insecure. Here we implement DNSSEC under the .pr country code top-level domain. To do so, we acquired the domain name luis.est.pr and a Virtual Private Server from HostGator.com with full root access. A PHP: Hypertext Preprocessor script was developed to aid in the process of periodic zone maintenance required by the protocol. Our resulting configuration was validated with three online tools used to test a successful DNSSEC environment.*

Key Terms *— BIND, DNS, DNSSEC, Security.*

BACKGROUND

The Internet has come a long way since its origins in the 1960s. To reach another computer on a network you need to know its unique number identifier called an IP address. There is one problem though, human beings are not as efficient at remembering number as computers are. To overcome this limitation, unique computer hostnames were assigned and mapped to their corresponding IP addresses. This way a hostname could be translated into an IP address which computers understand before a connection is established. Every hostname to IP mapping was listed as an entry in a HOSTS.TXT file shared across every user of the early Internet. As the Internet grew, also grew the size of the HOSTS.TXT file to the point where it became unpractical to continue distribution of such a large file. Because of its size and with so many new users rapidly joining the Internet, there were times when someone wasn't even finished downloading the file when there were already new entries being added to it giving rise to data accuracy issues. Another solution was needed and it came in the form of the Domain Name System (DNS). The DNS is basically a distributed database of host information that uses a tree or hierarchical name structure similar to the Unix filesystem and it translates hostnames like www.pupr.edu into their corresponding IP address [1]. At the top of the tree is the root node followed by the Top-Level Domains (TLDs), then the Second-Level Domains (SLD) and any number of lower levels, each separated with a dot [2]. Each node in the tree has a text label, which identifies the node relative to its parent such as google.com, facebook.com, yahoo.com, etc. Programs called nameservers contain information about some segments of the DNS database in zone files and make that information available to clients [1].

Data flows through the DNS following these steps. First, a zone file is created on the master DNS server. As stated earlier, this file contains all the DNS related information of the zone it will serve. Let's say this example zone is pupr.edu. Once the zone file is created it is very common for records to be dynamically added, deleted, or updated. These changes are automatically pushed into the master server which will then transfer the zone file to its slave servers to aid in the name resolution process. Up to this point, a zone administrator is responsible for these configurations. Now a caching forwarder is able to query either the master or any of the slaves for a DNS record. With everything in place, when you type in your browser “pupr.edu” a program called a

resolver will ask for the IP address of that domain on the DNS structure. It may ask a root DNS server, a recursive DNS, a caching forwarder, the master, or any of the slaves but it is entitled to give you an answer. And here is the thing: *any answer*.

Unfortunately, the DNS protocol was not designed with security in consideration and as a result it has several vulnerabilities that are exploited by malicious attackers. For example, when a computer asks a DNS server for the IP address of a hostname it will blindly accept any answer it receives as long as it is correctly formatted. Figure 1 illustrates this case showing that something has intercepted the traffic in the network between the computer and the correct server providing an incorrect answer [3].

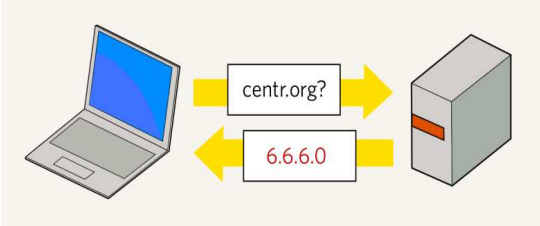

Figure 1
DNS Wrong Answer

Another common DNS exploit takes advantage of how quickly an answer is provided. Again, when a computer asks a DNS server for the IP address of a hostname it does not care where that answer comes from. Because of this, a strategically placed malicious DNS server could answer quicker than the correct server [3]. The computer will take this answer as valid. Figure 2 illustrates this scenario.

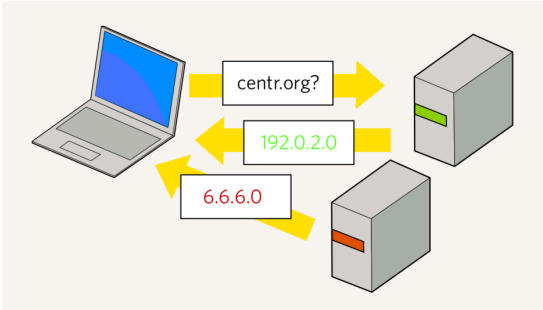

Figure 2
Quicker Wrong Answer

To address some of these vulnerabilities a set of extensions to DNS were developed by the Internet Engineering Task Force. These set of extensions are called the DNSSEC protocol and provide a way to verify data integrity and authenticity [4]. RFCs 4033, 4034, 4035 define the protocol and are colloquially called DNSSEC.bis [4]-[6]. In a correctly configured DNSSEC environment the exploits mentioned above simply cannot happen. We saw in the first case that something in the network could have been intercepting or redirecting traffic which resulted in a wrong answer provided to the resolver. This is not possible with DNSSEC because it uses a set of public/private cryptographic keys to ensure that a DNS answer has not been tampered with along the way. Any attempt to spoof or corrupt any response will be immediately detected and a *bogus* response will be returned instead [2]. The second exploit that we presented showed that a malicious DNS server responded faster with incorrect information. DNSSEC introduces a mechanism called the "Chain of Trust" which ensures that when you get a DNS response it is coming from an authorized server and not someone else. By taking advantage of the hierarchical structure of the DNS, the Chain of Trust can verify the authenticity of each node that answers a query. Of course additional information needs to be sent along in order to do this. That piece of information is called a signature. A DNSSEC aware nameserver will compare each answer against a list of known good signatures. If a signature mismatch is detected, an error will be returned. This works fine except that we run into the problem of scalability of the early Internet. Maintaining a list of known good signatures for every DNSSEC aware nameserver does not scale. A very elegant and simple solution could be achieved by starting validation with the top node of the DNS which is none other than the root zone. On July 16th, 2010 the root zone was officially signed allowing every domain on the Internet to implement DNSSEC in a more efficient scalable way [7]. Now instead of a list of known good signatures, only the root zone signature was needed for the Chain of Trust validation. This is not to say that DNSSEC did not existed prior to the root being signed. Without a root signature to start validation from the

top, small “islands of security” (fragments of the DNS tree) could enable DNSSEC with a mechanism called DNSSEC Lookaside Validation (DLV). With a fully functional signed root zone we will not elaborate on DLV.

Here is an example of how the Chain of Trust validates traffic. A DNSSEC aware resolver wants to know the IP address of the domain pupr.edu and it has the root zone's signature. It will ask a root server and will verify that the signature matches the one it has stored. The root answers with the address and signature of the .edu domain. Finally, it will ask the .edu server for the IP address of pupr.edu and will verify that the signature provided by the root matches the one answering as the .edu server. If it does, we now know that the IP address of pupr.edu could have only came from an authenticated server.

Puerto Rico has its own place in the DNS structure. It resides within a section called the Country Code Top Level Domains (ccTLD) and it has its own two letter identifier (.pr) assigned by the Internet Assigned Numbers Authority (IANA) [8]. This project consists of implementing DNSSEC under the .pr ccTLD.

PROCEDURE

First we needed to register a .pr domain name. The Gauss Research Laboratory, Inc. is the company that manages the .pr ccTLD (NIC.PR) and is offering free domain names with the .est.pr extension to students. Upon evaluating the available extensions listed at their website (www.nic.pr) it was an obvious decision to choose and register the domain luis.est.pr for this project. Second, we needed a dedicated server to act as the authoritative nameserver for the zone luis.est.pr in which DNSSEC will be implemented. For this we turned to HostGator.com and considered the products they offered. Although a dedicated server would be ideal, they are much too expensive for our budget starting at $139.00 monthly. Instead we settled for an alternative solution that works just as well, a fully managed Virtual Private Server with CentOS, 1.13 GHz CPU, 768 RAM, and 30 GB of space for $39.95 a month. The package also came with two fixed public IP addresses: 198.57.133.93, and 198.57.137.204. We used the first one for the VPS. Next we installed the most widely used nameserver software in production, the Internet Systems Consortium's BIND. With the VPS up and running and with BIND installed we proceeded to create the zone file for luis.est.pr. It contained "A record" entries for the domain's nameservers ns1.luis.est.pr with IP address 198.133.57.93 and ns2.luis.est.pr with IP address 198.133.137.204.

To make the zone file accessible we needed the .pr nameservers to point to it. This was done by creating the ns1.luis.est.pr and ns2.luis.est.pr entries in the DNS section of our account at NIC.PR. In about 24 hours the information was replicated through the DNS and the nameservers ns1.luis.est.pr and ns2.luis.est.pr answered authoritatively for the domain luis.est.pr.

With correctly configured DNS servers hosting the zone file for luis.est.pr we are now ready to implement DNSSEC following the guide described on [2]. There are four steps that must be completed for a successful DNSSEC implementation.

1) Generate a cryptographic key pair.
2) Add keys to zone file.
3) Sign the zone.
4) Edit BIND's configuration file.

Here we present the details of each step for the domain luis.est.pr. DNSSEC uses two cryptographic keys to provide authenticity and data integrity to the DNS. The first one is the Zone Signing Key (ZSK) and it is used to sign all the records in the zone file. The second key is called the Key Signing Key (KSK) and it is used to sign only the key records of the zone. The ZSK will sign everything including itself and the KSK. The KSK will only sign itself and the ZSK. To generate the ZSK we first changed to our working directory /var/named/run-root/var/ and executed the following command in a single line:

```
# dnssec-keygen -a DSA -b 768 -n zone luis.est.pr
```

The dnssec-keygen command creates a 768 bit key with the Digital Signature Algorithm. It produces two files Kluis.est.pr.+003+18204.key

and Kluis.est.pr.+003+18204.private. As the name implies, the private file should be kept securely and offline. To generate the KSK we executed the following command:

```
# dnssec-keygen -a DSA -b 768 -f KSK -n zone luis.est.pr
```

This completes the first step. Now we need to add the generated keys to the zone file which can be done in a number of ways but we chose to use the *cat* command. The *cat* command will append the contents of the .key files to the end of the luis.est.pr zone file. These are the commands we executed:

```
# cat Kluis.est.pr.+003+18024.key >> include.luis.est.pr
```

```
# cat Kluis.est.pr.+003+33510.key >> include.luis.est.pr
```

Continuing with the third step it is now ok to sign the zone. Signing the zone means attaching digital signatures to the existing DNS records of the zone file. To sign the luis.est.pr zone we executed the following command on a single line:

```
# dnssec-signzone -o luis.est.pr -t -g -k Kluis.est.pr.+003+33510.key luis.est.pr Kluis.est.pr.+003+18024.key
```

The dnssec-signzone creates a new file with the .signed extension. This new zone file contains the DNSSEC signatures. The -g flag generates a file named dsset-luis.est.pr that contains a Delegation Signer (DS) record. This file was sent to the .pr ccTLD to include it in the est.pr zone file. Once the DNS administrator of the .pr ccTLD updates the est.pr zone with our DS record we would have successfully joined the DNSSEC Chain of Trust.

The last step in order to make DNSSEC work is to tell BIND to serve the luis.est.pr.signed zone file instead of the regular one. To do that we edited BIND's configuration file (named.conf) and added the lines marked in red:

```
options {
        dnssec-enable yes;
};
zone "luis.est.pr" {
        type master;
        file "luis.est.pr.signed";
};
```

All that is left to do is to restart the nameserver with this command:

```
# service named restart
```

DNSSEC has now been successfully implemented for the domain name luis.est.pr. Although everything is in place and we have a working configuration, DNSSEC introduces a time factor that is not present in regular DNS. Zone signatures have a default expiration date of 30 days. If the zone is not resigned before the expiration date is met then DNSSEC will stop working and *bogus* answers will start to show. To avoid this problem a PHP-CLI automated script was developed and configured to run daily as a Cron job and increment the SOA record serial number. Since our VPS came with Crontab preinstalled we only needed to put our script on the /etc/cron.daily directory for it to execute every day. At the end of execution it calls a second script that we created that resigns the zone luis.est.pr.

Every time the zone resign script is executed, the dnssec-signzone command will not have the -g flag because we have already provided the .pr ccTLD with our DS record.

RESULTS

The goal of this design project was to implement DNSSEC under the .pr ccTLD. We used three different tools to validate our configuration: an online DNSSEC checker from Surf Net [9], a web browser add-on developed by CZ.NIC [10], and the dig (domain information groper) command querying the open DNSSEC validating resolvers from OARC [11]. Figure 3 below displays the results of Surf Net's Live DNSSEC Checker.

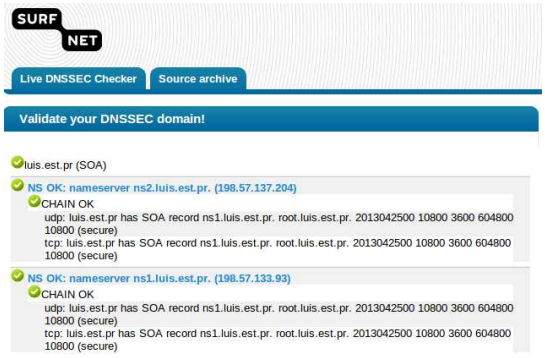

Figure 3
Surf Net's Online DNSSEC Checker Results for luis.est.pr

This online tool queries the SOA record of luis.est.pr and verifies that the DNSSEC Chain of Trust is valid from the root node down to luis.est.pr. You can see that both nameservers respond with *secure* which means that DNSSEC is working properly. The second tool used for validating our setup was the DNSSEC Validator add-on for Mozilla Firefox displayed in Figure 4. The results can be appreciated next.

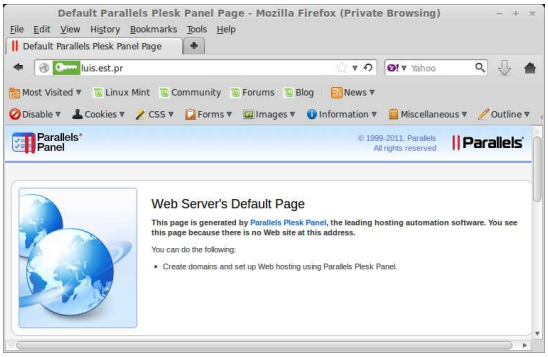

Figure 4
CZ.NIC's DNSSEC Validator Browser Add-On

To the left of the URL, the add-on displays a key icon over a green background if DNSSEC is working correctly which can be clearly seen on Figure 6 next to luis.est.pr on the address bar. For this validator to work as expected it is necessary to change its settings and configure it to use CZ.NIC's open DNSSEC resolvers. The third mechanism used to validate our DNSSEC configuration was to query directly the open DNSSEC resolvers from the Domain Name System Operations Analysis and Research Center (DNS-OARC). Here is the dig command output of said query.

```
medina@medina-pc ~ $ dig +dnssec
@149.20.64.20 luis.est.pr
; <<>> DiG 9.8.1-P1 <<>> +dnssec
@149.20.64.20 luis.est.pr
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status:
NOERROR, id: 55417
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2,
AUTHORITY: 3, ADDITIONAL: 5
```

The dig command response has been shortened for brevity but the important fact to notice here is the *ad* flag – short for Authenticated Data – displayed on the header section of the answer shows that DNSSEC is working correctly.

We are proud to discover that this is the first domain to adopt DNSSEC under the .pr ccTLD which had DNSSEC available since 2006 even before the root zone was signed. This implies that to the time of this writing not even the Government of Puerto Rico has taken benefit of the security improvements that DNSSEC offers. In an insecure environment such as the Internet, DNSSEC provides the much needed aspects of authentication and data integrity desired by any service or web site such as online government agencies that want to provide users with a more secure channel of communication of directory lookups [12]. In fact, on August 22, 2008, the U.S. Department of Homeland Security mandated that every domain under the .gov zone should deploy DNSSEC by 2009 [13]. According to the NIST's Information Technology Laboratory up to April 26, 2013 only 82% of the U.S. Government domains have successfully implemented DNSSEC as depicted by Figure 5 below [14].

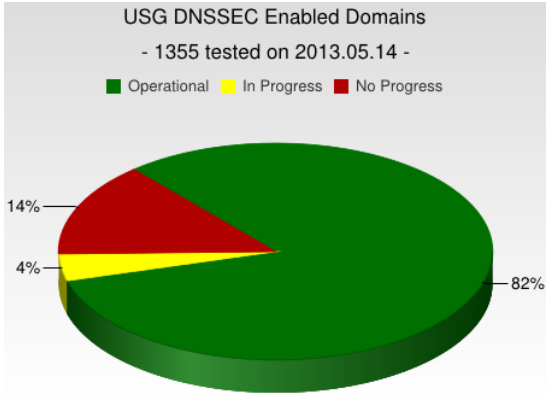

Figure 5
U.S. Government DNSSEC Deployment

The road to a successful DNSSEC implementation is not without a few problems that must be overcome. With regular DNS once you configure the nameserver software you can mostly set it and forget it because it is a onetime task. On the other hand, DNSSEC signatures have a default

expiration date of 30 days after which if the zone is not resigned stale and bogus data will be served and no validation will occur. To avoid this situation we developed a script that runs daily and automatically resigns the zone. However, before resigning the zone it is crucial to increment the zone's serial number in the SOA record. Ignoring to do so will not propagate the new signatures to the secondary nameservers. To overcome this, a PHP script was created to read the current serial number and increment it by one before calling the zone resigning script. Of course there was also the prerequisite of acquiring an affordable fully managed server in which to host the zone file but that was solved by turning to virtual private servers that offer the same functionality.

An improvement to our configuration and a recommendation as future work would be to use NSEC3 instead of NSEC. You might have noticed that we did not mention NSEC on this report. That is because it is the default option when signing a zone. NSEC stands for Next Secure and it is a resource record used by DNSSEC to provide authenticated denial of existence of a record. This might expose a vulnerability in which an attacker can "walk the zone" by following the chain of NSEC records and thus listing the whole zone contents. NSEC3 provides the same denial of existence but does not allow such vulnerability [15]. Also because of our VPS underwhelming specifications we could not generate larger keys. It is recommended to generate keys of larger size (e.g. 2048 bits) and choose a stronger algorithm such as RSASHA256.

SUMMARY

We have showed the benefits that a DNSSEC environment can bring to any online infrastructure. However, we also recognize that understanding and subsequently implementing the protocol could impose a learning curve that may be deemed too steep for some to even care. After many years of hard work, the Internet community has taken the necessary steps to increase network security by signing the root zone. With this work we hope to have demystified misconceptions about the hardships of adopting DNSSEC by establishing a self sustained implementation from the ground up.

REFERENCES

[1] Albitz, P, *et al.*, "How Does DNS Work?", *DNS and BIND*, 2006. 13-25

[2] Aitchison, R, "DNSSEC", *Pro DNS and BIND*, 2005. 283-328.

[3] Davies, K, “DNS Cache Poisoning Vulnerability Explanation and Remedies”, Internet Assigned Numbers Authority, 2 Mar. 2013, <http://www.iana.org/about/presentations/davies-viareggio-entropyvuln-081002.pdf>

[4] Larson, M, *et al*. “DNS Security Introduction and Requirements.”, The Internet Society, 10 Mar. 2013, <http://tools.ietf.org/html/rfc4033>

[5] Larson, M, *et al*. “Resource Records for the DNS Security Extensions.”, The Internet Society, 19 Mar. 2013, <http://tools.ietf.org/html/rfc4034>

[6] Larson, M, *et al*. “Protocol Modifications for the DNS Security Extensions.”, The Internet Society, 21 Mar. 2013, <http://tools.ietf.org/html/rfc4035>

[7] “Status Update, 2010-07-16”, *Root DNSSEC*, Internet Corporation For Assigned Names and Numbers & VeriSign, 3 Abr. 2013, <http://www.root-dnssec.org/2010/07/16/status-update-2010-07-16>

[8] “IANA — Root Zone Database.”, Internet Assigned Numbers Authority, 5 Apr. 2013, <http://www.iana.org/domains/root/db/>

[9] “DNSSEC Checker Version 1.0.16”, SURF NET, 12 Apr. 2013, <http://www.dnssecmonitor.org>

[10] “DNSSE Validator”, CZ.NIC, 22 Apr. 2013, <http://www.dnssec-validator.cz>

[11] “OARC's Open DNSSEC Validating Resolver”, Domain Name System Operations Analysis and Research Center, 23 Apr. 2013, <https://www.dns-oarc.net/oarc/services/odvr>

[12] “DNSSEC – What Is It and Why Is It Important? | ICANN.”, Internet Corporation For Assigned Names and Numbers, 23 Apr. 2013, <http://www.icann.org/en/about/learning/factsheets/dnssec-qaa-09oct08-en.htm>

[13] Evans, K, “Memorandum for Chief Information Officers”, Executive Office of the President of the United States, 25 Apr. 2013, <http://georgewbush-whitehouse.archives.gov/omb/memoranda/fy2008/m08-23.pdf >

[14] “Estimating USG IPv6 and DNSSEC External Service Deployment Status”, NIST Information Technology Laboratory, Advanced Network Technologies Division, 26 Apr. 2013, <http://fedv6-deployment.antd.nist.gov/snap-all.html>

[15] Blacka, D, *et al*. "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", The Internet Engineering Task Force Trust, 4 May. 2013, <http://tools.ietf.org/html/rfc5155>