

Incident Response Guide to the Kaminsky DNS Cache Poison Exploit

Team Cymru

Fall, 2008

Abstract

Recently a major DNS (domain name system) vulnerability was discovered affecting one of the most critical components of the Internet infrastructure. This newly developed exploit takes advantage of an inherent weakness in the DNS as it is widely deployed on the Internet. As we will show, due to the distributed nature of the DNS, the vulnerability poses a significant challenge in ensuring the safety of the Internet for users and organizations worldwide. Anyone with malicious intent could threaten the availability or trustworthiness of the Internet for millions, causing significant disruption, confusion and financial loss. We outline mitigation challenges and provide a brief summary of tips that may help responders if they become victim to a DNS cache poisoning attack.

Key Take-aways

- The DNS exploit disclosed by Dan Kaminsky is real and is a serious threat.
- While many users and organizations have patched, there will be significant portions of the Internet susceptible to this exploit for many years.
- A successful attack could potentially redirect all web, email and other Internet traffic away from a victim user or organization. This may result in a variety of problems including denial of service, phishing, eavesdropping, extortion, fraud and identity theft.
- Detection of attack attempts, successful or not, can be difficult. The attack traffic can be made to blend in with otherwise normal-looking traffic.
- Tracking back an attack to a perpetrator may be very difficult. Attack traffic is forged to appear to come from somewhere other than where the real attacker is.
- If this vulnerability becomes codified into common malware or becomes widespread as an attack vector, it is likely to cause significant disruption, data loss and great financial risk for some time while the industry scrambles to patch and deploy mitigation mechanisms.

- There are multiple independent demo versions of the exploit with source code freely available.
- There is a growing community of support and tools available to help mitigate and respond to an attack.

1 Introduction

The domain name system (DNS), a critical component of the Internet infrastructure, has recently been the subject of a serious exploit discovered and disclosed by Dan Kaminsky, Director of Penetration Testing for IOActive Inc. This exploit takes advantage of an inherent weakness in the DNS as it is widely deployed in the Internet. While mitigating the vulnerability is possible, the fix must be widely deployed to be most effective. However, the DNS is highly distributed and large parts will likely remain vulnerable for many years. From large ISP servers to small consumer hardware devices that attach home users to an ISP with a DSL or cable modem connection, an overwhelmingly significant portion of the Internet community may be at grave risk. In addition, detecting an attack and uncovering the perpetrator may be difficult without a serious commitment by law enforcement, ISPs, information security companies, DNS software vendors, and users. We do not believe the community is well situated to adequately respond to this threat vector.

In this guide we first address the inherent problems of mitigating the vulnerability, paying specific attention to hindrances an investigation would likely incur in response. We detail common mitigation strategies and suggest additional avenues of opportunity to thwart the most devastating types of attacks. For those needing a refresher, Appendix A describes the some of the relevant component of the DNS. The nature of the exploit discovered and disclosed by Dan Kaminsky is summarized in Appendix B.

2 Why This Exploit is Significant

Implicit in the use of, the size of, and the distribution of the DNS is the intrinsic need for widespread safeguards. This newly discovered exploit has the potential to cripple or even subvert Internet services for many users and organizations on the Internet. Government, business, academia, and consumers are all potentially at grave risk. An attack can target multiple disparate portions of the Internet, which makes mitigation highly problematic since there is no one component that needs to be protected, but instead all components must be protected, even ones for which you are not responsible. Figure 1 shows how widespread open caching resolvers are throughout the entire Internet address space.

Furthermore, this vulnerability could be exploited in a stealthy fashion, causing damage for the target name or end user with them unaware that they are even being victimized. Its as if someone walked off with your purse or wallet at lunch time and you didn't even notice it was missing until the following

morning's breakfast. This causes problems not only for mitigation, but also for response.

There are numerous scenarios one can envision where a DNS cache poison attack may be used. We will try to enumerate just a small sample to help demonstrate the sort of risk associated with this vulnerability.



Figure 1: Summer 2007 Map of Open Resolvers, courtesy Duane Wessels

2.1 Owning the Internet

While this threat in and of itself cannot take down the entire Internet per se, it could seemingly take down the Internet from the perspective of a user or organization that is relying on vulnerable caching resolvers. An attack could poison all of the TLD name servers, and since there is very little usable data above the TLD for an average client, that is in effect the entire Internet, or at least the entire named Internet. If Internet communications used only IP addresses and never issued DNS queries then the threat would be non-existent, but that is impractical scenario today.

2.2 Phishing Attack

Perhaps an attacker wanted to only steal credentials for a user account or specific site. A phishing site could be launched, impersonating the real destination server. Then an attacker poisons the name of that site in a vulnerable caching resolver. An unsuspecting user might then be tricked into entering their credentials on the fake site. The attacker may even forward the credentials to the real site, redirecting them after successfully intercepting the credentials.

2.3 Host Exploitation

Rather than redirect a victim to a phony site, an attacker could force a victim to a particular site that hosts another exploit, perhaps one that will become active when the victim connects or tries to render whatever it was they were trying to do in the first place. Drive-by downloads, or malicious web pages, may activate when a vulnerable web browser or browser plug-in visits.

2.4 DDoS

An attacker could poison a name so that it renders the intended answer unattainable. Alternatively, an attacker could poison a popular name so that it points to another site unprepared to handle the increased traffic load, resulting in a distributed denial of service attack (DDoS) attack for both the poisoned name and ultimate the destination site.

3 Mitigation Challenges

There are a number of inhibitors to minimizing this DNS cache poisoning threat. These include:

- Lack of awareness, interest, or perceived threat.
- Belief that it is someone else's problem.
- Monitoring and troubleshooting tools are not widely deployed.
- DNS expertise is limited and not a common function for most IT staff.
- Auditing and logging infrastructure for the DNS requires significant resource investment.
- The DNS protocol in widespread use has no authentication nor verification enforcement mechanisms.

4 Mitigation Strategies

There are a number of means by which to mitigate this recent DNS threat. It is possible to make the attack impractical, but responders may still need some strategies to employ when a caching resolver is under attack. We outline a few brief strategies and ideas, many of which have been widely discussed elsewhere.

1. Evaluate the caching server infrastructure and apply upgrades and patches that help address the issue.

ISC BIND and Microsoft DNS servers platforms are two of the most popular name server implementations and both were found to be at significant risk. Figure 2 illustrates how little entropy and randomness there are in selection of resolver source ports, a critical weakness exploited by this recent threat. Ensure the most recent versions of your resolver are installed and running, many now add additional source port selection randomness.

2. Limit the accessibility of caching resolvers. If possible, do not allow just anyone to send them queries.

Caching servers accessible to the entire world are also known as open resolvers. While there may be good reasons for resolvers being open, it is safer if they are not. ISC BIND and other implementations have the concept of views, which permit restricted access to the resolver based on the source IP address. With a view properly configured recursion can be disabled for all, but a select set of sources. Disabling recursion helps thwart the attack by disallowing just anyone from forcing the resolver to perform iterative queries.

3. Enable detailed caching resolver logging and statistical gathering. If possible, keep a record of all queries sent to a caching resolver for at least a week.

Keeping a history of DNS server statistics and especially query history can consume a significant amount of resources. If possible, send server logs to a remote logging server and run log summary and analysis tools on the logging server. Some implementations of the syslog daemon in UNIX systems may make heavy use of the CPU, especially if query logging is being performed. Alternative syslog daemons such as `syslog-ng` may be a good alternative in those situations. Another option is to capture raw data or DNS messages as they traverse the network to or from the caching server using a dedicated packet capture server.

4. Befriend key DNS software implementers and service providers. Build up a relationship with the people on the front lines.

Get to know your software vendors. Join DNS operator user groups and mailing lists to keep abreast of the latest issues and strategies.

- Investigate technologies such as DNS Security Extensions (DNSSEC), understand how they work, who is deploying them and what the strengths and weaknesses are.

DNSSEC is an IETF-sponsored add-on to DNS that attempts to provide DNS message authentication and integrity assurance. While it has been in development and testing for many years, interest has increased.

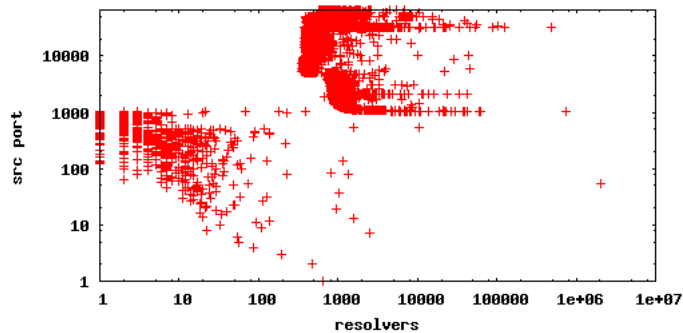


Figure 2: Resolver Source Port Distribution

5 Tools and Resources

There are a number of DNS tools and resources that can help mitigate and respond to DNS threats such as the recent cache poison vulnerability. Here are just a few we think you should know about.

DSC

The DNS Operations and Analysis Center (DNS-OARC) makes a the DNS Statistics Collector (DSC) freely available. This tool can be run on or near any DNS resolver or server. DSC captures DNS packets and summarizes statistics by particular query and answer fields. More information about DSC can be found at <https://www.dns-oarc.net/oarc/src/dsc>.

DNS Entropy Tester

The DNS-OARC also makes available a web and command-line based tool that estimate the relative entropy a caching resolver has, which directly effects the difficulty level of an attacker being able to commit a successful cache poison attack. This tool can be found at <https://www.dns-oarc.net/oarc/services/dnsentropy>.

Zonecheck

This tool interrogates a domain or name servers looking for anomalies, configuration problems or other issues that may be of concern related to your DNS presence. You can freely submit your zone or name server addresses into a web forum and get back your results in real-time here <http://www.zonecheck.fr>.

Public DNS Server and Zone Monitors

There are a handful of organizations that perform monitoring on key components of the DNS namespace and infrastructure, often making available summary data. RIPE has a DNS monitoring service at <http://dnsmon.ripe.net/dns-servmon/> and Team Cymru has a similar service at <http://www.team-cymru.org/Monitoring/DNS/>.

Log Summarization

There are a number of log parsing and summary tools. LogReport <http://www.logreport.org> is a generic log reporting tool that has support for some DNS implementations. One of the Cymru also has a BIND specific log parsing called named-report tool that summarizes a host of BIND-specific data. It can be found at <http://www.cymru.com/jtk/code/>.

DNS-Operations Mailing List

The dns-operations mailing list is an email list dedicated to discussing the topics of operating the DNS. Many DNS experts from across the globe participate, but it is open and accepting of newcomers. To subscribe, visit <https://lists.dns-oarc.net/mailman/listinfo/dns-operations>.

Open Resolver Test

The Measurement Factory has an easy to use web-based submission form that can determine if a resolver is publicly accessible or not. This is helpful to determine if a particular resolver one is using is open and potentially more susceptible to attack than one that is closed. It has maintains a history of open resolvers it finds in periodic surveys. For additional information visit <http://dns.measurement-factory.com/cgi-bin/openresolvercheck.pl/>.

6 Incident Response First Steps

If you suspect a given server or organization have been subject of a DNS cache poisoning attack, some of the steps you may wish to follow to perform triage and investigate the incident include the following:

1. Document the date, time and timezone each step of the way.

2. If you're working from a UNIX shell, keep a history of all the troubleshooting steps you take. One easy way to do this is by using the `script` command, which will send all STDIN and STDOUT to the file name that is passed as a parameter to the script command.
3. If you are using a tool such as `dig`, be sure to carefully construct the query. Specify the server to query against (`@` parameter), the query type (e.g. `A`, `ANY`, `CNAME`) and whether you want your query to have the recursion bit set or not (`recurs` flag).
4. If you have access to the caching resolver system under investigation, take a snapshot of the cache. In BIND, you can issue the `rndc dbdump -cache` command. Be sure to save old copies of the `named.dump.db` file before running this command.
5. If you do not have access to the caching server, be sure you know exactly which server is in question. Sometimes a caching server may be configured in an anycast group. Note the source address from where suspect queries are coming from. If you are doing remote troubleshooting using `dig`, this would be the host you run `dig` from. Run a `traceroute` to the caching resolver to get an idea of where in the network it is located. Some servers may support a `dig @caching-server ch txt hostname.bind` query that may help identify a specific instance of a caching server being queried in an anycast group.
6. Contact a partner or group that you trust if you need additional assistance. Know your limitations and don't be afraid to request help. Especially since cache poisoning attacks may persist for only a short while. Time may be of essence and you do not want to miss an opportunity to uncover a root cause or perpetrator.
7. If you know a name is poisoned and you must clear the poison to fix a critical issue, flush the cache or reload the server as a last resort, but try to save a copy of the cache before doing so.
8. Some vendors and ISPs have DNS DDoS mitigation devices that may help. While a cache poison may not be causing a DDoS per se, these special devices can be configured to force all DNS queries to switch to TCP. This is done relatively transparently to the DNS querier and server and helps thwart DNS-based spoofing attacks since now the attacker must be able to complete a TCP session, which is much more difficult to spoof.
9. Setup a packet capture or enable DNS query logging if you suspect a poison attempt. It may take some time to analyze the data, but having that data will be invaluable in a postmortem investigation.
10. Do not panic. Sometimes a cache poison isn't really a cache poison. An authoritative server could be mis-configured or compromised. Be careful not to jump to a conclusion too quickly.

7 Evidence Preservation

Here are some additional considerations if you own or operate a DNS caching server that comes under attack. Obtaining and preserving data that may help identify key characters and potential lead to a suspect is of utmost importance.

- Capture MD5 or SHA-1 hash signatures of critical system and log files of any system involved in an exploit.
- Dump system and DNS cache memory.
- Know which source IP addresses can send recursive queries to the caching resolver and force iterative queries to be made.
- Copy all relevant DNS server log and configuration files to read-only media.
- Seal a copy of the read-only media and give it to attorney for safekeeping.
- Document everyone who has physical and remote login access to systems, document network topologies, IP addresses and other relevant configuration information.
- Many systems are monitored by other systems on the same network. These other systems, though not directly involved in an attack, are often where a great deal of the evidence may reside. Be sure to include them in your discovery and evidence collection.
- Know who and when to consult with for technical guidance. Know your limitations and seek assistance when necessary.

A DNS Technology Refresher

The domain name system, or DNS, provides a simple and convenient way to arbitrarily associate a common name to an Internet resource. So for example, The White House is a common name associated with the building located at 1600 Pennsylvania Avenue in Washington D.C. Likewise, the name `www.team-cymru.org` is a name in the DNS that is associated with the Internet Protocol (IP) address `68.22.187.6`. Names in DNS can be associated with many types of information and we have shown just one common example, the IP address. Consumers, businesses, governments and other entities all rely on a properly operating DNS to locate Internet resources. However, no one entity is responsible for all the world's DNS service. DNS, like the Internet, is a distributed system where many players have competing incentives and interests, all of which are geographically and commercially dispersed. This leads to widespread disparity in the operation and reliability of the DNS, which at times serves to one's advantage, but in some instances may pose a challenge. The recent DNS threat that forms the basis of this paper is one such case where

the widespread diversity of the DNS poses a significant challenge to the safe operation of the Internet.

A.1 DNS Namespace

There is an agreed upon hierarchical structure to the naming scheme used by today's Internet DNS. This hierarchical structure, referred to as the DNS namespace, even though distributed throughout the globe, does contain some centralized and concentrated points of control. The naming hierarchy purposely consolidates authority to select providers and trustees at the upper levels of the hierarchy. A detailed study of the namespace hierarchy is not essential to our understanding of DNS and this current threat, but it does provide insight that may suggest certain mitigation and monitoring tactics we have outlined.

Every name in the DNS is a string of characters separated by dots (.). The strings between the dots are called labels. Labels can be made up of almost any character except dots. Names are case insensitive and some systems restrict the character set to a subset of the modern English alphabet characters, numbers and a dash (-). The protocol specifications permit other characters, but they are less common and there are some additional rules that have been instituted in specific implementations of the DNS over the years. Implicit in a name is a trailing dot, but it is rarely written. For example, `www.example.com.` is written with the trailing dot, but many people and systems just use `www.example.com` for convenience. The top, or apex, of the hierarchy actually begins at the rightmost edge of a name with that trailing dot whether it is written or implied. As we traverse the name hierarchy (from right to left) we add labels separated by dots.

The boundaries signaled by the dot separator are referred to as zones or containers. So for example, `team-cymru.org` is a zone. In a zone you can have host names or names that refer to child zones. A child zone is what it sounds like, another layer in the hierarchy. So take for example the name `marketing.example.com`. `marketing` may identify a child zone within the `example.com` parent zone. Within `marketing.example.com` you might have the names `www`, `mail` and `ads`. Each of these prepended in `marketing.example.com` may be a complete host name by itself or they may be parent zones that contain another level of names to Internet resources. A web server specific to the `ads` located in Chicago for example may have the fully qualified domain name of `www.chicago.ads.marketing.example.com`.

The upper layers of the namespace hierarchy are determined and allocated by central authorities, but as we traverse through the namespace, zones are often controlled by increasingly local organizations, administrators, or even individuals.

Within at least the first two levels of the hierarchy there is significant diversity of responsibility for the maintenance of the namespace and the DNS servers. Reading names right from left, the very first label is the top level domain (TLD). For example, `com` is the top level domain in `www.example.com`. Naturally `example` is the second level domain (SLD). However, in this context

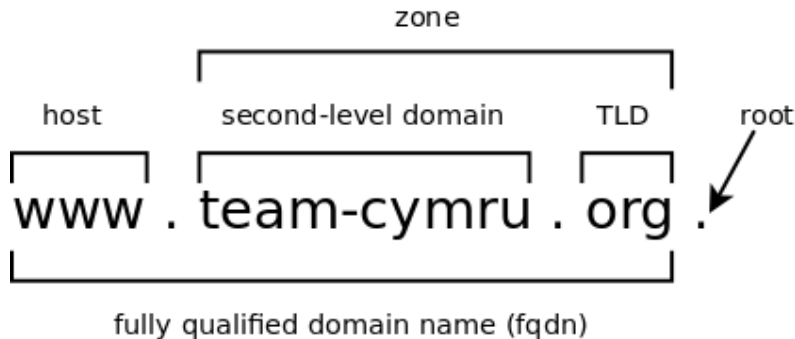


Figure 3: Anatomy of a Domain Name

`www` is not a third level domain (3LD). Instead, the name terminates at this left edge and we say `www` is a host in the `example.com` zone. The entire name string `www.example.com` is called a fully qualified domain name (FQDN). The Internet Cooperation for Assigned Names and Numbers (ICANN) is ultimately responsible for coordinating the assignment and management of names near the top of the hierarchy. ICANN and the namespace managers at the upper levels of the namespace hierarchy help ensure that the delegation of authority for child zones and names flows naturally and reliably while being interoperable between DNS providers and users. We will explore more about how this namespace is administered a little later, and one can learn more about ICANN from their website at <http://www.icann.org>.

A.2 Protocols

DNS messages are encapsulated within the user datagram protocol (UDP) or the transmission control protocol (TCP), both of which are encapsulated within the Internet Protocol (IP). By and large, most DNS traffic is UDP-based. UDP is simple and enables application multiplexing on top of IP. That is, it enables multiple applications to run over the same source and destination IP address pair. So for example, host A may use DNS with host B while at the same time host A can use the trivial file transfer protocol (TFTP) to host B. This will work, because UDP will differentiate these two different application sessions using a unique set of *port numbers* between the two hosts. These port numbers are determined at the UDP layer and are what permits application multiplexing between a pair of communicating hosts. Without these port numbers the two hosts in our example will not be able to easily differentiate what traffic is intended for the DNS application and what traffic is for TFTP.

UDP is lightweight, trivial to implement, and easy to troubleshoot. However, it can pose a challenge, particularly when security is of concern. DNS is a just a request/response protocol and when using UDP there is a simple two-packet

exchange. A client sends a query packet and the server sends back an answer. There is no precondition to generating a DNS message other than properly formatting it for the application to understand. There is no authentication in UDP and DNS as they are commonly deployed. Since many DNS servers are intended to provide public, Internet-wide service, they often have little to no access control limitations on the service. This openness and the two-packet UDP message exchange for DNS can facilitate spoofing attacks. For law enforcement, this could pose a significant investigative challenge.

The attack this paper discusses requires an attacker to be able to send forged answer packets. For better or for worse, many networks around the globe permit hosts to spoof their source IP address. Therefore, traditional network flow data, or simply flows, may be of little use in discovering the true origin of an attack. Additional insight is likely needed to successfully mitigate and respond to an attack against a vulnerable configuration.

A.3 DNS Message Format

A DNS message takes on one basic format with various fields utilized or filled in depending on whether the message is a query or an answer. The type of question (or answer) will further dictate details of the message. There are numerous fields defined in a standard DNS message, but the basic format and structure can be found in Internet RFC 1035. Please note that RFC 1035 document is fairly old and there have been some updates to the specifications as well as some unwritten implementation intricacies that can alter the behavior or format of DNS messages in practice. With the exception of a brief sidebar into DNSSEC, we will not delve further into more precise definitions or the latest enhancements.

There are five parts to a DNS message, the header, the question, the answer, the authority and the additional section.

The header is mandatory for both queries and responses. It contains fields that define the rest of the DNS message, capabilities requested by the client and a 16-bit identifier field used to match returned answers to outstanding queries. As we will soon see, often this ID field is all that stands in the way of an attacker and this latest DNS threat.

The question contains three fields, one of which is effectively obsolete so we will ignore it. One field is the QNAME or simply the query name that is being looked up. It is the fully qualified domain name, the labels separated by dots. Another is the QTYPE field. QTYPE is used to restrict the query to the type of information the client is interested in. For example, one type is an A query, which asks for an IPv4 address that is associated with the QNAME.

The remaining three parts, the answer, authority, and additional section, all have the same format, but are used for different purposes. These sections will generally only be present in response packets.

Each of those parts consists of zero or more resource records (RRs). A resource record is essentially an entry in the DNS database. The DNS database contains records all having the following attributes: NAME, TYPE, CLASS, TTL, RDLENGTH and RDATA. The NAME is the fully qualified domain name

to which the rest of the data in the record pertains. The TYPE defines the meaning of the RDATA portion of the record such as the A or address type we just mentioned. The CLASS is effectively obsolete and rarely used in practice. The TTL is a numerical value representing the time in seconds that this record may be cached for before having to be discarded. The RDLENGTH denotes the length of the RDATA field. Finally the RDATA is the information that describes the resources. Note, there may be multiple records with the same NAME, but different values in all or some of the other fields. If there are multiple RRs with all the same fields except for the RDATA attribute, then as a group those names are said to be part of the same RRset.

If the DNS message is a response and contains an answer, one or more RRs will be put into the answer section. An authoritative section may contain the names of the authoritative servers for the answer. The additional section generally contains the A RRs for the authoritative servers that may be listed in the authoritative section.

A.4 DNS Clients and Servers

There are three basic types of DNS network software components in widespread use: the authoritative server, the caching resolver and the stub resolver. There are some subtle differences between each, which we will explore in the following sections. In addition, some components may be combined or further dissected into other sub-classes. We will briefly describe just the three basic types.

A.4.1 Authoritative DNS Server

DNS names and their associated data must be populated into the DNS database somehow. For example, Team Cymru populates the database by configuring a set of name servers with names and associated data in or under the `team-cymru.org` zone. This can be done manually or through an automated process, but at some point the data gets put into authoritative servers and is made available to the rest of the network. While it doesn't matter how the authoritative data gets entered into the system or even precisely which systems are deemed authoritative, the act of loading that set of data is done through the authority of the zone owner. In other words, Team Cymru is the zone owner responsible for ensuring the data in the `team-cymru.org` zone is loaded, correct and available. One or more authoritative servers are solely responsible for answering queries for names in the configured zone(s).

A.4.2 Caching Resolver

A caching resolver, also known as a full resolver, sends queries to authoritative servers or even sometimes to other caching resolvers. Often times caching resolvers act on the behalf of others, usually end systems like PCs, workstations, or other end user devices. If possible, a caching resolver retains a local copy of any answers to questions it receives for as long as the TTL associated with the

RRs allows. This caching function prevents the resolver from having to repeat queries it can otherwise retain locally even if only on a temporary basis. This is an optimization that can be very beneficial from a performance and reliability standpoint, but as we will see it is this caching function coupled with answer spoofing that is at the heart of this recent DNS exploit.

A.4.3 Stub Resolver

Most end stations such as PCs, printers and other single-user devices tend to use what are called stub resolvers. A stub resolver is a programming interface between applications and caching resolvers. Typically end stations will obtain a set of nearby caching resolvers with which they can utilize. Whenever the end station needs to perform a DNS lookup, it can contact one of these caching resolvers and ask it to perform all the work of traversing the namespace on its behalf. Stub resolvers tend to have limited functionality and rely on caching resolvers not only to do all the work, but to also maintain a cache. Therefore stub resolvers may often repeat questions frequently.

A.5 Domain Name Registrations

For the DNS database and namespace to be populated there must be an organizational structure that coordinates getting data into the system at the upper levels. While control is distributed and zone owners can configure and manage their own names and servers, people do have to agree on the apex of the namespace and pointers from parent to children zones. With the guidance of institutions such as ICANN there are registries, registrars and registrants, which form the cooperative to populate the DNS database and make it available to the Internet.

A.5.1 Registry

A registry is responsible for one or more top level domains (TLDs) in the DNS namespace. So `.com`, `.net`, and `.us` are real world TLDs, and each has a single registry operator associated with it. A registry may operate multiple TLDs, but there is only one registry per TLD. A registry ensures not only proper operation of the managed TLD zone, but is also responsible for running a whois service for the zone. Whois is another protocol, separate, but in this case related to the DNS. It is an informational directory not unlike a whitepages for DNS names. For every name in a TLD there is an associated set of information about that name including the designated authoritative server, contact information for the name owner, registration data, when the last changes were made and a pointer to the registrar who is responsible for this whois record. The whois database and the TLD zone itself, while run by the registry, are actually populated with data provided to them by registrars.

A.5.2 Registrar

Registries maintain the authoritative DNS database, the DNS servers and whois service for the TLD they are responsible for, but they do not deal directly with the name owners in most cases, particularly for the generic top-level-domains (gTLDs) such as .com, .net, and .org. Instead, accredited registrars are the interface to the end customer. Registrars are essentially domain name brokers who provide the interface to the registry service. While in some TLDs the registry and registrar are the same entity, usually they are not.

A.5.3 Registrant

A registrant is a domain name holder or owner. A registrant can control their own DNS zone within one of the many TLDs or other levels usually by paying a recurring fee to a registrar. A registrant is then assigned an account through a registrar who collects your zone information, contact details, name server configuration and possibly zone data if they are going to provide authoritative service on their behalf. Contact details are published in the whois directory service. For a variety of reasons many registrants prefer not to publish their details so they often subscribe to some sort of *privacy* registration service. In this case the registrant technically transfers control of the name to a proxy, a third party who acts as the contact on their behalf and is technically responsible for the name. In practice the privacy service will proxy all instructions for zone configuration changes and contact updates through to the registrar.

A.6 DNS Lookup Example

We will now examine Figure 4 and the narrative that follows to gain a basic understanding of how the DNS works.

In Figure 4 above we see an end user, John Doe, on his computer at home attempting to visit his bank's web page. Perhaps Mr. Doe might be reviewing his checking account balance or maybe he is setting up automated bank payments for his telephone bill. Before a single banking transaction can take place, John Doe's computer first performs a DNS lookup for the `www.example.com` name he has typed into his browser. More precisely, after entering the bank web site name (`www.example.com` in this example), his computer issues a request onto the network to locate the Internet Protocol (IP) address for the bank's web server. This IP address is a little like a mailing address, such as the 1600 Pennsylvania address we mentioned at the outset. In some cases it may work to type in this IP address instead of the name, but John Doe and most users find it much easier to use a more intuitive reference like `www.example.com`. In this sense DNS provides the helpful mnemonic for accessing Internet services by name rather than a more cumbersome string of digits.

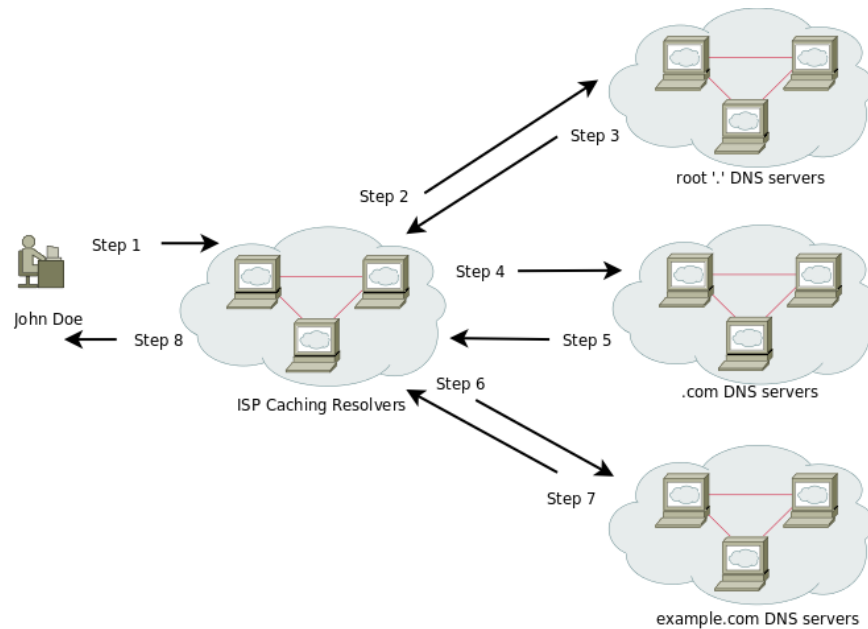


Figure 4: Caching Resolver in Action

Step 1

John Doe's PC, which is running a stub resolver, sends a lookup request to one of the local ISP's caching resolvers. We will assume the selected caching resolver has just come online and has no entries in its cache. The only thing the caching resolver knows about is a permanent list of root servers hard coded into every caching resolver installation throughout the world. This list of 13 IP addresses represent the top of the DNS hierarchy, that rightmost dot often implied at the end of every fully qualified domain name. We use the root as an entry point into the global Internet DNS namespace.

Step 2

Once the caching server receives the question from the stub resolver, it checks its cache and, finding nothing, sends the query to one of the root servers to start its iterative search through the namespace.

Step 3

A root server won't know the answer to this question. All it knows about are name servers associated with TLDs. So instead of an answer it sends back a referral. A referral is like a hint or a signpost, indicating someone else to ask who may have better information. In this case the referral will be to the .com

DNS servers. Note, the caching resolver will retain this hint for as long as the TTL in the referral message specifies.

Step 4 and Step 5

Similar to queries sent to the root, the com name server will give a referral to the `example.com` name servers. Again the caching resolver remembers this information and continues on its way, closer and closer in anticipation of an answer.

Step 6 and Step 7

The ISP caching resolver contacts one of the `example.com` servers in step 6 and asks for the IP address associated with `www.example.com`. At last the caching resolver has reached an authoritative server for the name `www.example.com`. An `example.com` name server returns an answer to the ISP caching resolver in step 7,

Step 8

The ISP caching resolver, caches the answer and forwards it on to the stub resolver and John Doe in step 8. Process complete.

Lookup Summary

Note, at any point in time, the name may not have existed or there could have been a typo. In that case an error would eventually be returned to the application. Alternatively, if a packet was lost or delayed, a timeout mechanism might kick to, resulting in an error to the DNS process or application.

Also note that many other users and systems similar to John Doe and his PC may be using the ISP caching resolvers simultaneously. As a caching resolver builds up a cache, the frequency of having to go through all steps in the process above is minimized. Most end systems tend to use a caching resolver provided by their host organization or ISP, but some individuals can and do run a full caching resolver on their system rather than a stub resolver. This can actually provide some additional level of protection from this DNS threat, because it may remove a vulnerable caching resolver from their resolution path.

Finally, it is noteworthy to consider the sheer number of accessible caching resolvers on the Internet. These are caching servers that are open and accessible to anyone on the Internet. They will answer queries from anyone who sends them one. This isn't always a bad thing, as there are some providers who purposely provide this service. However, many of these open resolvers are not well managed and often not going to be trustworthy in situations such as this. Unfortunately, many people use these open resolvers for one reason or another. Some surveys have shown there to be millions of open caching servers. Consider the extent of this DNS threat knowing that there are potentia

B The Kaminsky DNS Cache Poison Attack

It may be helpful to review the DNS Refresher section if you haven't already before reviewing this section. A thorough understanding of how the DNS works will be necessary to understand the recent threat and vulnerability disclosed by Dan Kaminsky. We mentioned elsewhere that answer spoofing and caching, which are fundamental to the vulnerability, but before going any further we need to first understand what a DNS cache poison is and what the essential avenue is for a DNS cache poison to occur.

A cache poison occurs when an incorrect answer to a question finds its way into a DNS cache. A wrong answer may not even be noticeable to most users depending on what the name and wrong answer turn out to be. If however the name is one that is important to someone, a wrong answer could result in denial of service attack, identity theft, eavesdropping, fraud or other security breach. A cache poison could be relatively benign, impacting no one, or it might ruin the day for a large group of users or organizations. The size and scope of the attack is limited by two things: one, the user population that relies on the caching resolvers, and two, by the motivation and skill of an attacker. Fewer users translates into fewer problems as a percentage of the Internet user base. An attacker may wish to poison a name that is not critical to the user base or it could be to send all email to the attackers inbox.

This vulnerability relies on the ability to spoof two things. One, the IP address of an authoritative server that is queried by a vulnerable caching resolver. Two, the answer to a question the caching resolver asks an authoritative server. Remember steps 6 and 7 in Figure 4. There the caching resolver asked a question of an authoritative server for `www.example.com`. What if instead of receiving an answer from a real `example.com` nameserver, the ISP caching resolver instead received a fake answer from someone else? How is that possible? What would the fake answer need to get right for the ISP caching resolver to accept it?

B.1 Authoritative Server Spoofing Requirements

To successfully spoof a DNS answer, here are some things that need to be considered:

Source IP Address

You need to know the address of the authoritative server you're pretending to be. If the ISP caching resolver is widely accessible, you can generally force the caching resolver to query any set of authoritative server you want. It may only query one of the set, but the set size is small, thirteen or less, and usually two. Difficulty: EASY.

UDP Source Port

You need to know which UDP source port the ISP caching resolver sends queries from. The caching resolver must include a source port in the UDP/TCP portion of the message. This is so the authoritative server will know to which port to deliver the answer in a response. If the ISP caching server is already widely accessible, this is often trivial to discover for most resolvers, because they tend to use the same source port for all queries. Difficulty: EASY.

DNS QNAME

You need to not only know the authoritative server, but you need to know what question the ISP caching resolver is going to ask the authoritative server. If the caching resolver is widely accessible, you can force this to be a name of your choice. Difficulty: EASY.

DNS Query ID

Each query a caching resolver generates gets a 16-bit ID associated with that specific question. This is used to help associate outstanding queries with returning answers. 16 bits provides up to 65,535 different values, which is a fair amount. Most implementations also do a pretty good job of randomizing this value on each successive query and if the answer does not have a matching ID value the answer is discarded. Difficulty: CHALLENGING.

Response Time

From the time the ISP caching resolver asks the authoritative server a question until it receives a response is about how long an attacker has to try to send a correct spoofed answer. An attacker does not get penalized for sending wrong answers; they can send as many wrong answers as they want. If they eventually get one through that matches the expected parameters, the attacker wins. This might slightly reduce the DNS Query ID spoofing difficulty. However, it really depends on network conditions. Difficulty: EASY.

B.2 Attack Example

As you might have guessed, the vulnerability Dan Kaminsky discovered improves the attacker's chances of success significantly. It takes advantage of all the weaknesses presented above, including weak spoofing protection and the luxury of being able to return incorrect guesses without being punished.

Imagine an attacker wants to poison the name `www.example.com`. Imagine the legitimate A RR points to `192.0.2.1`, but the attacker wants to change it to `192.0.2.2`. Using the technique disclosed by Dan Kaminsky we will explore how this attack can be amazingly effective on a vulnerable caching resolver.

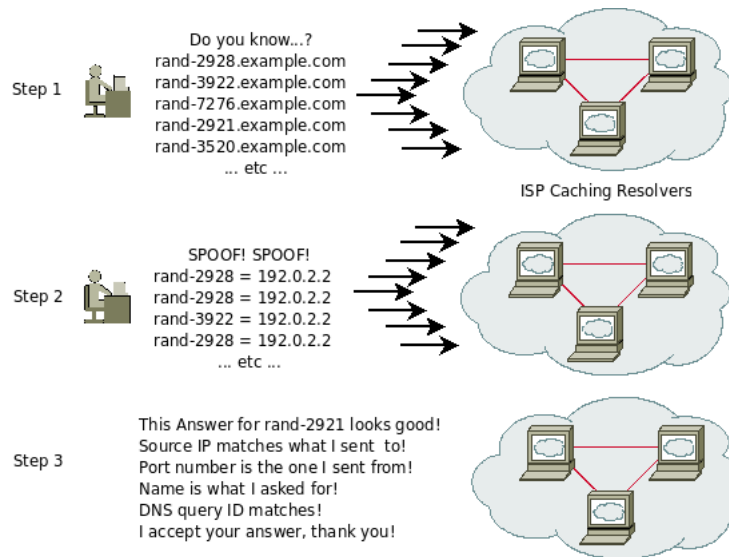


Figure 5: Kaminsky Attack in Action

Step 1

In Figure 5 an attacker sends legitimate looking queries for random names in the `example.com` zone to a vulnerable caching resolver.

Step 2

An attacker can immediately send forged answers for those queries to the vulnerable caching resolver. In fact, the attacker doesn't just send a single forged answer, an attacker can send multiple answers. Remember, the attacker might have to guess at some of the values the caching resolver expects in the answer, particularly the correct DNS message ID, which is probably the least easy thing to discover.

Step 3

Eventually the attacker will guess the correct ID value and the spoofed answer will be accepted by the caching resolver. Keep in mind the attacker is very likely to eventually return a correct answer before the true authoritative server can. The attacker can send spoofed answers immediately after sending a query, while the caching resolver has to wait for its query to be delivered to the real authoritative server, be processed, and a response returned. The caching resolver will accept the first matching answer so the attacker has a big head start. Dan showed that this can occur within just a few seconds based on the above techniques.

From the previous steps we know that we were spoofing an answer to a random name in the zone, but that by itself isn't help to an attacker. An attacker wants to poison a name that will be of some value to the caching resolver or to users of the caching resolver. The one last trick is in the answer that is being spoofed. The answer is for a delegation to the name an attacker wants to poison. What the answer says is, I don't know what this answer is, but here is a referral, go visit `www.example.com`. So the answer is a referral, but there is an A RR in the additional section. It tells the caching server how to contact `www.example.com`. If you read the DNS Refresher section you may remember when we discussed how during the DNS lookup process the caching resolver caches referrals. This is just another instance of caching a referral, but in this instance the caching resolver is tricked into thinking that the `www.example.com` is a name server and caches the IP address given to it by the attacker.

Glossary

3LD

3rd-level domain.

authoritative server

A DNS server configured with local information for one or more zones and provides definitive answers to questions about data contained in those zones.

cache poison

See poison.

child

In a namespace hierarchy, a child inherits the parent zone and is delegated authority for a namespace within the parent zone.

container

See zone.

delegation

A cut in the namespace that defers authority for a child zone to another authoritative name server set.

DNS

Domain Name System, a distributed database that maps names of resources to information about those resources.

DNSSEC

DNS Security Extensions. A set of specifications designed to provide authenticity and integration to the DNS.

domain name

The locator used in DNS, represented by a series of character strings called labels. Labels are separated by dots, which create the hierarchical structure to the name.

flows

Summarized network traffic data, primarily header information identifying source, destination and traffic characteristics, but not content.

forwarder

A caching resolver that queries and caches on behalf of other resolvers.

FQDN

Fully qualified domain name.

full resolver

A resolver that is capable of performing both iterative and recursive queries for itself or on behalf of others as well as caches answers according to a TTL.

gTLD

Generic top level domain, a subset of TLDs that include `.com`, `.net`, `.org` and others.

host name

A domain name that refers to a specific host.

iterative query

A resolver that performs all queries necessary itself, following delegations as necessary to reach an answer. A recursive or caching server typically performs iterative queries on behalf of stub resolvers.

label

A string of characters that identify a node in the namespace. The label may be a domain container or a host name in a zone.

lookup

See query.

name server

A TCP/IP server system that responds to DNS queries. There are various classes of DNS name server systems depending on the configuration and implementation.

namespace

The hierarchical structure, from right to left, of label strings, separated by dots, that associate various attributes to Internet resources.

open resolver

A DNS querier that permits anyone from anywhere to direct it to perform iterative queries without restriction by simply asking.

parent

In the namespace hierarchy, a parent is an ancestor zone or nameserver set to the child, a child inherits the parent zone.

poison

An incorrect answer to a DNS query stored in a caching resolver.

query

A type of DNS message, used to ask for a mapping between a domain name and something about that name.

recursive query

A resolver that requests another resolver to perform any additional queries necessary on its behalf to resolve a name. A stub resolver typically issues recursive queries.

recursive server

DNS server that performs iterative name lookups on behalf of other clients, usually stub resolvers.

referral

An answer that points to another name server set in the namespace. Ideally the referral points the querier one step closer to an authoritative answer.

registrant

A name owner who has a relationship with a registrar to ensure details about their name and contact details are entered in the TLD and whois database.

registrar

A name broker who provides the interface between registries and registrants.

registry

A TLD zone operator.

resolver

A DNS component that performs queries. There are varying classifications of resolvers, including minimal resolvers such as stub resolvers and full-featured resolvers called full resolvers.

RR

Resource Record, an entry in the DNS database that consisting of the name, class, record type, value and TTL.

SLD

Second-level domain.

spoof

A fake; in DNS, a DNS message with fraudulent identifiers, query data and/or answer data.

stub resolver

A simple message passing interface used by end devices such as PCs, provides interface between applications on a host and caching resolvers in order perform name lookups.

TLD

Top-level domain.

TTL

Time-to-live, the maximum time (expressed in seconds) that a caching resolver should store an answer it receives from another server.

zone

A container for names and delegations to child zones at a particular point in the namespace hierarchy.

References

- [1] Dan Kaminsky, *Its the End of the Cache as We Know It*, Black Hat 2008, http://www.doxpara.com/DKM_B02K8.ppt
- [2] Ryan Singel, *Feds Start Moving on Net Security Hole*, Wired Blog Network, October 8, 2008, <http://blog.wired.com/27bstroke6/2008/10/feds-take-step.html>
- [3] Steve Friedl, *An Illustrated Guide to the Kaminsky DNS Vulnerability*, August 7, 2008, <http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html>
- [4] A. Hubert, R. van Mook, *Measures for making DNS more resilient against forged answers*, Internet-Draft, August 14, 2008, <http://www.watersprings.org/pub/id/draft-ietf-dnsext-forgery-resilience-07.txt>
- [5] T. Grance, K. Kent, B. Kim, *Computer Security Incident Handling Guide*, NIST Special Publication 800-61, January 2004, <http://csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf>
- [6] Technical Working Group for Electronic Crime Scene Investigation *Electronic Crime Investigation: A Guide for First Responders*, U.S. Department of Justice, Office of Justice Programs, July 2001, <http://www.ncjrs.gov/pdffiles1/nij/187736.pdf>
- [7] B. Collie, *Collecting and Preserving Evidence After a System Compromise*, AUUG Security Symposium 2000, <http://mangrove.nswrno.net.au/dist/public/auugsec2000/Collecting%20and%20Preserving%20Evidence%20after%20a%20System%20Compromise.ppt>
- [8] M. Braid, *Collecting Electronic Evidence After a System Compromise*, AusCERT, August 02, 2001, <http://www.auscert.org.au/render.html?it=2247>
- [9] P. Mockapetris, *Domain Names - Implementation and Specification*, IETF RFC-1035, November 1987, <http://www.ietf.org/rfc/rfc1035.txt>