Recursive DNS Cache Auditing

Prepared by: Jose Avila III
Edited By: Brett Watson

July 25, 2008

ONZRA 16068 Maricopa Highway Ojai, CA 93023 **T** 805-201-8905 jose@onzra.com www.ONZRA.com

Introduction

Dan Kaminsky of IOActive recently discovered a flaw in multiple DNS server implementations. The flaw is detailed in US-Cert Vulerability Note VU#800113. Dan coordinated efforts with multiple experts, including Paul Vixie, to organize what could arguably be one of the most responsible disclosures conducted in this industry. This disclosure and subsequent release involved multiple vendors meeting together to discuss issues, and courses of action, and then simultaneously releasing patches to the public.

In the months leading up to the public release, Jose Avila of ONZRA discussed the need for an open source solution for detecting cache poisoning events with Dan. As a result of that discussion, ONZRA developed *CacheAudit,* which ONZRA released under the BSD License.

Now that details of the exploit have been leaked to the Internet through various blogs, it is more critical than ever to patch as soon as possible. Even when (if?) this issue is resolved there may be more cache poisoning flaws in the wild, as yet undiscovered. As Dan said in his blog, he "could have missed something." Until DNS security measures are in place (can you say, "DNSSEC"?), organizations need to be on their toes. As such, ONZRA hopes this project will aid organizations in detecting wether or not they have been compromised.

This white paper provides a brief background on DNS as it relates to cache auditing, a method for auditing recursive DNS server caches, and the details of *CacheAudit,* which is based on these auditing methods.

Background On DNS

We cover two types of DNS servers: recursive and authoritative. Authoritative name servers only serve the records that they are authoritative for and refer non-authoritative recursive queries to the root servers. Most clients never query the authoritative servers directly, but rather query a local recursive server. This recursive server will query the authoritative name servers when needed.

In contrast, recursive servers will answer both authoritative, and non-authoritative queries, and will store answers it receives in memory. This is known as caching. The answer will be cached for the length of time specified by the authoritative name server TTL, or Time To Live. When a recursive server receives a query, it will first check to see if it has an answer to return to the client from cache. If it does have a valid response, it will return that response to the client. If a response does not exist in cache, the recursive server will iteratively query the roots, and subsequent authoritative servers, to find the proper answer to return to the client.

The Cache Dump

Most modern resolvers have the ability to dump an existing cache to a file. This file contains the name, RR type, remaining TTL, and answer for each cached record. Given that there is little performance impact in

dumping the cache, and that this can be done while the name server is running without affecting normal operations, we have a good means to perform additional processing of the cached data externally.

If there were no changes being made in DNS, and there were not multiple views of a zone from different physical locations, in theory one should be able to re-query the same name servers and get the same answers as those in the cache dump. Given there are many different configurations for DNS, and that DNS is an ever changing environment, the validation procedure gets a bit more complex, but that's what we are here to tackle.

Validating The Constantly Changing Cache

When validating a recursive server's cache, we not only want to know if the cache is accurate at a particular point in time, but we want to know that it is always accurate. When things are not accurate, we want to know. This can be done by regularly taking cache dumps (say every 5 minutes) from a recursive name server and validating them. There is no point in re-validating something you validated during the last iteration, and as such, with each new iteration, you can compare against the previous iteration, and validate the changes. These changes would either be new entries, or modified entries in the cache. The validation process contains three basic steps: 1) Validation against authoritative name servers, 2) Validation against peers that also validate against authoritative servers, and 3) Validation of Content Deliver Networks (CDN).

Authoritative Validation

Once you have the set of records to validate, the validation service must query to find the authoritative name servers for those records. Once the authoritative name servers are found each one is queried, and the results are compared against the cached recordset in question. If the results returned from the authoritative name server match the cached recordset, that particular cached entry has been validated. If the contents of this recordset were not validated at this stage, it may indicate a true cache poisoned record, but it's also possible the record could have been legitimately modified at the authoritative server since the last time the cached entry was made.

Peer Validation

Peers are other recursive DNS servers within your infrastructure that use a different cache than the server you are validating. These recursive servers are referred to as peers as they should be servers that are part of your infrastructure so as to not place an additional load on other providers' resources.

If we are only validating against the authoritative name servers, the record may have changed since the time the dump was processed. Since recursive DNS peers cache an answer for the specified time to live provided by the authoritative name server, there will be a historic representation of what was served by the authoritative name servers in the recursive cache. As the cache of one of the peers could have been poisoned, there should be a required threshold of agreement among recursive peers before acknowledging that an answer is valid.

Content Delivery Networks (CDN) Validation

When testing initial versions of the *CacheAudit* script, one deficiency noted was the vast number of false positives. A good portion of these were from Akamai. As we queried repeatedly for *n3g.akamai.net*, we would receive a different set of IP addresses for each query. The sets of IP Addresses that were returned would vary based on the source address of the query. (See Table 1)

California	Illinois
64.215.162.85	207.65.73.227
64.215.162.47	207.191.178.194
64.215.162.21	207.191.178.194
64.215.162.77	207.65.73.227
64.215.162.21	207.65.46.161
64.215.162.79	207.191.178.194
64.215.162.37	207.65.46.161
64.215.162.52	207.65.46.161
64.215.162.60	207.65.46.161
64.215.162.5	207.65.46.161
64.215.162.4	207.65.46.161
64.215.162.71	207.65.46.161
64.215.162.38	207.65.73.227

California	Illinois
64.215.162.69	207.65.46.161
64.215.162.30	207.65.73.227

Table 1.

From analyzing the results in Table 1, we assumed that if queries originate from the same location in California, anything in the 64.215.162.0/24 range is a valid response, whereas if queries originate from Illinois, anything in the 207.65.46.0/24, 207.65.73.0/24, and 207.191.178.0/24 ranges would be valid.

Through testing we only found up to 3 class C networks utilized for a particular record. As long as our validation is happening from the same physical location as the resolver cache that we are analyzing , this has proven to be a reasonable method for lowering the false positive alert ratio.

Alerting

If a record set has not been validated in the authoritative, peer, or CDN validation stages, an alert should fire. It is important to note, when alerts are fired it does not mean that your server for sure has been cache poisoned, but rather there has been a discrepancy. This record should be investigated further to determine weather or not it is due to an attack. In the event that false positives or actual poisoning events are detected, ONZRA would be interested in the details as to better improve future revisions of this product.

CacheAudit currently logs entries for events to the local syslog server. If you are using Linux or OSX this should not be a problem. ONZRA plans to support email notifications in the future. As *CacheAudit* is open source, the application can easily be extended to support other alert mechanisms.

Using *CacheAudit*

CacheAudit currently supports Microsoft DNS, BIND, and PowerDNS. Each of these resolver implementations have a mechanism for dumping the in-memory cache. You can use *netcat* to forward the requests to the *CacheAudit* service on your network.

BIND

```
rndc dumpdb; cat /var/named/data/cache_dump.db | nc <host> <port>
```

Microsoft DNS

```
nc -v -t -e "dnscmd.exe /ZonePrint ..Cache" <host> <port>
```

PowerDNS

```
rec_control dump-cache /tmp/cache; cat /tmp/cache | nc <host> <port>
```

Administrators may want to note that if using the PowerDNS, or Microsoft DNS recursive server, the server will queue, and temporarily not answer queries. If BIND 9 is compiled with threads, it will not lock, and will continue to answer queries during the cache dump process. The duration of time it takes to dump the database may vary based on size of the cache, and processing power. With MIcrosoft DNS it is recommended that the data get dumped locally and then transfered across the network, as opposed to being dumped from a remote machine. As with deploying any software in your infrastructure, ONZRA highly recommends testing in a testbed prior to deploying throughout your network.

Summary

The recursive DNS provider community needs more tools like this to assist in detecting and troubleshooting cache poisoning events. While this tool is in its early stages, I hope that feedback from the community will help drive the design of future revisions, so we can better arm ourselves in the future for more events like those discovered by Dan Kaminsky. If there are any modifications that are needed to get *CacheAudit* working in your environment, or if you would just like to request additional features, feel free to contact ONZRA.

About ONZRA

ONZRA focuses on highly specialized security solutions and industrial grade development consulting services. ONZRA has strategic partnerships with many of the top security and development groups on the planet. These are the best of breed, industry movers and shakers. You have read their books and used their tools; you may have even seen them speak at conferences around the world like Black Hat, RSA and ORAC.

From providing Cisco security design and training, architectural product development, vulnerability assessments of your web presence and web applications, to guiding your in-house development team towards a launch date, ONZRA is a one stop shop for the absolute best of the best.

Mailing Address
16068 Maricopa Highway
Ojai, CA 93023
Direct: 805-201-8905
Email: Jose.Avila@ONZRA.com
www.ONZRA.com