Patching Nameservers: Austria reacts to VU#800113

CERT.at Report #2

www.cert.at <team@cert.at>

Otmar Lendl <lendl@cert.at>

L. Aaron Kaplan <kaplan@cert.at>

July 24, 2008

Table of Contents

Executive Summary

This paper analyses the impact of the coordinated efforts to patch Austria's recursive DNS server infrastructure following the revealings of Dan Kaminsky (US-CERT VU#800113) which showed that almost all DNS servers on the Internet are vulnerable to DNS cache poisoning[1]. CERT.at – being run by NIC.at, the Austrian domain registry – is in a special position to be able to assess the reaction of the Austrian nameserver operators to the discovered DNS vulnerability. We analyzed the rate at which DNS servers were patched from an insecure to more secure state. The paper discusses a methodology to measure the patch level "score" of a recursive DNS server. We believe that this score methodology can be applied to cleanly discern patched from unpatched DNS servers.

We describe a methodology how a TLD operator can use his query logs to check which operators have patched their DNS resolvers according to the published advisories.

The conclusions are rather grim so far – **more than two thirds** of the Austrian Internet's recursive DNS servers **are unpatched** while at the same time the upgrade adoption rate seems rather slow. Our findings are matched by the observations of Alexander Klink of Cynops GmbH[2] who analyzed the results of the online vulnerability test on Dan Kaminsky's doxpara[3] site.

We hereby present the information to the concerned public in the hope that DNS – a central and crucial part of the Internet – remains secure.

Our recommendation to IT system administrators is to update their recursive DNS servers immediately and check[4] that their upgrades were successful.

Motivation

On July 8th, 2008, almost all vendors of DNS resolver software announced updates to their software to protect the integrity of DNS caches against a new kind of attack. This synchronized release was arranged by CERT-CC to give the public about a month to protect themselves before details of the attack are scheduled to be released.

Given the almost unprecedented coordination amongst the CERTs and the vendors, and the language in the advisories it had to be assumed that the integrity of the DNS is indeed in grave danger. CERT.at thus issued a strong recommendation to its constituency to implement the suggested countermeasures as soon as possible[5].

CERT.at is run by nic.at, the Austrian ccTLD Registry. We thus have a close affinity to issues related to the DNS infrastructure, and decided to use some of the nic.at resources in order to assist our constituency.

First, we used the nic.at communication channels to alert all ISPs and registrars to the threat.

Secondly, operating the authoritative nameservers for ".at" gives us a view into the behavior of all resolvers with some affinity to .at, which maps nicely to our constituency as national CERT for Austria.

Based on our findings we will alert our constituency to update their DNS servers.

1 http://en.wikipedia.org/wiki/DNS_cache_poisoning
2 http://www.shiftordie.de/articles/Traversing%20Dan%27s%20directory
3 http://www.doxpara.com/?p=1162
4 http://member.dnsstuff.com/tools/vu800113.php
5 http://cert.at/warnings/warnings/20080708.html

Background

The attack

Initially the details on the attack were not public. However the international security community was able to recreate Dan Kaminsky's attack. The following key facts were already contained in the advisories.

The attacker needs to:

- Get the target to resolve queries for him.
- Forge answers from the real authoritative nameservers.
- Guess various parameters to get its forged packets to be accepted as correct answer.

For the last part, the attacker must get the following fields right:

1. Source IP address (of the authoritative nameserver)
2. Destination IP address (from which the query was sent)
3. Destination UDP port number (from which the query was sent)
4. The Query itself (QNAME, QTYPE)
5. Query-ID (a 16 bit field)

All this is well known and documented e.g. in [ID-forgery-resilience[6]]. The following fields used to be easy to guess: 1. (can only assume a handful of values), 2., 3., (by getting the target to send a query to a server under control of the attacker) and 4., leaving only the 16 bits of the Query-ID as the defense against forgery.

This is clearly not enough randomness given today's commonly Internet bandwidth which allows sending 2^{16} packets within a few seconds.

The countermeasure

One of the proposed countermeasures is to randomize the source port from which the resolver sends queries to authoritative nameservers. This has the potential to add another 16 bits of randomness which the attacker has to guess, making a successful attack much more difficult.

This “source port randomization” (SPR) has been implemented by all major DNS software vendors. Any patched resolver should thus never use the same source UDP port for all his queries.

As source port randomization changes the active behavior of nameservers (and not, as typical security fixes do, change how software reacts to certain inputs), monitoring the query patterns of resolvers shows immediately whether this resolver implements SPR.

Observing the deployment of SPR patches

The constituency of CERT.at as the national Austrian CERT is the Austrian Internet. Usually, such a

6 http://tools.ietf.org/html/draft-ietf-dnsext-forgery-resilience-05

national CERT has no means of checking (let alone enforcing) whether its advisories were followed. Our situation as part of nic.at gave us the chance to survey the patch-discipline within our constituency and target reminders at exactly those parties which have not fixed their servers.

The following facts work to our advantage:

- We have access to query-logs of .at ccTLD nameservers.

  To our further advantage, no changes were necessary: these logs already contain the source UDP port number for each query. We thus have enough data to establish a baseline of resolver behavior before the vulnerability was announced.

  There was no need to look into the queries content, only source IP address and source UDP port number are needed.

- Resolvers in Austria are likely to query for domains under .at.

  The Austrian TLD ".at" is the default TLD for Austrian businesses, and even international Businesses have registered their .at domain (and if just for a web-forward to their .com) as this is what Austrian users are expecting.

- Interesting targets make a lot of queries.

  As a first estimate, the more users rely on a resolver, the more interesting it is for an attacker to insert a spoofed record into its cache. While some enthusiast might install his own full resolver on his home machine, most Internet users will simply use his ISP's servers. Poisoning the cache of a private box does not have the broad effect of an successful attack against an ISPs resolver. Such a big ISP resolver will query for a large number of .at domains, thus leaving enough entries in our logfiles to assess its query patterns.

  While a highly targeted attack against a corporate resolver (which might not show up in our analysis due to too few queries) is possible, such resolvers should (hopefully) only recurse for a tightly restricted community which makes them harder to attack.

The basic idea

Given the query log of an authoritative nameserver for .at, find the Austrian IP addresses which sent the most queries (these are the relevant resolvers!) and analyze their source port distribution (do they implement SPR?).

Methodology

We restricted our analysis to the logfiles of a single .at nameserver located in Vienna. While this reduces the number of queries we see from each client, the most important clients still showed up in highly significant numbers.

Data reduction

The server we looked at receives about 35 million requests from roughly 750,000 different IP addresses per day, so that any analysis covering weeks of traffic risks being swamped by too much irrelevant data. We filtered by:

- Client IP address in Austria

  Based on geoIP / RIPE DB inetnum data. This reduces the number of different IP addresses down to about 10,000 per day.

- Ten or more queries in a daily logfile.

  As per reasoning above, we do not care about ultra-low volume resolvers. In addition, a low number of source port samples limits the statistical significance of any conclusions.

With these two limitations, we are in the range of 4000 IP addresses per day. As not the same resolvers make the cut each day, we found that over two weeks we accumulated data about 10,000 resolvers.

At this stage, we also had to exclude a few client IP addresses from the further analysis, e.g.

- nic.at internal monitoring queries
- Domain catchers looking for deleted domain
- Anything which looks like scripts iterating over a list of domain, and not normal resolvers.

Scoring Resolvers

For each client, we now have a series of ten or more source port numbers. Between the queries to our nameserver, that resolver probably queried hundreds of other nameservers. We thus cannot take our observed sequences as the one which the resolver generates for subsequent queries. A detailed statistical analysis on the randomness properties of these sequences is thus of little value.

Instead, we focus on two crude metrics which proved to differentiate nicely between unpatched and patched servers:

- Port changes.

  A simple count of changes of the source port number between subsequent requests. An unpatched server will change his query port only on restarts, thus registering hardly any port changes. Even if a resolver cycles only through a very small pool of 1000 different ports, the number of port changes will be close to the number of requests.

  This test can be fooled by simple NAT scenarios, e.g. when two unpatched resolvers behind a simple NAT will be seen as oscillating between two source port numbers.

- Number of different ports seen.

  Given a sufficiently large sample of queries, this should converge towards to number of different ports this resolver uses and thus gives an indication on the amount of protection added by SPR.

The simple NAT scenario from above will register on this test.

In order to get a single score (0= bad, 1=perfect) for each client, we use the following reasoning:

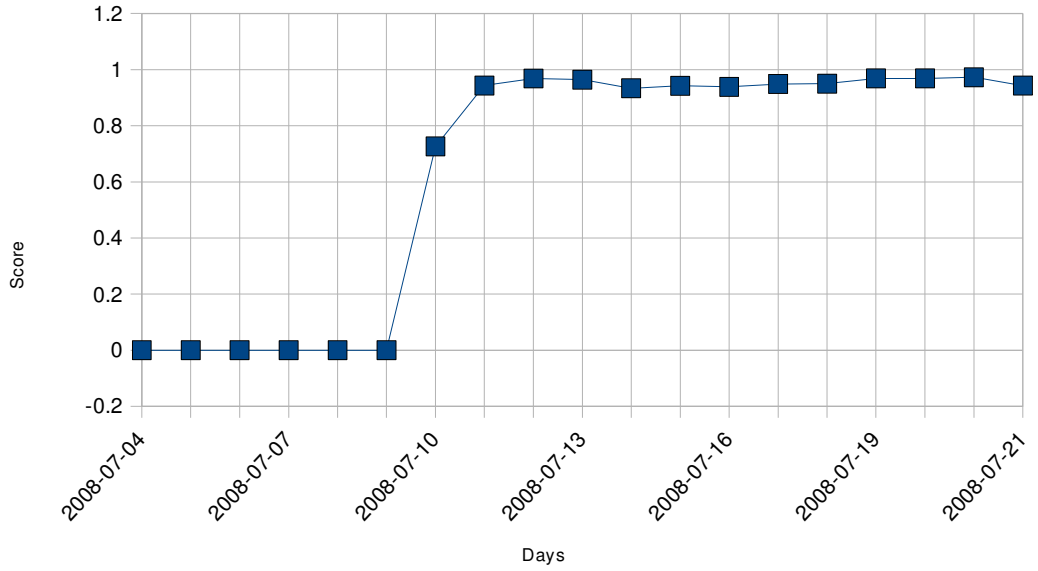
Regarding port-changes, using simply #portchanges / #queries does the trick: this ratio is always between 0 and 1 and resolver implementing SPR will get a number close to 1.

On the first sight, we could the same with #ports: the closer this is to #queries, the better. We cannot use that, as #ports is bounded by 65536, whereas #queries is not. Thus the formula becomes #ports / min(#queries, 65536).

A good resolver needs to do well in both test, thus a simple multiplication of the two results yields the final score for a resolver. We thus arrive at:

$$score = \frac{portchanges}{queries} * \frac{ports}{min(queries, 65536)}$$

We calculate now this score for each client for each day. Looking a some specific nameservers we find that this score works, as can be seen by the values for one ISPs nameserver:



Based on this graph it is easy to see that this resolver was patched on July 10th.

Limitations

- We do not evaluate the randomness in the source port selection.
- We do not look at the randomness in the query-ID at all.
- NAT devices can mess with our results.
- Restricting the client IP addresses to officially Austrian IP space might miss resolvers used by transnational ISPs.

Classifying Resolvers

The score value rates the behavior of a resolver on a single day. In order to determine whether a resolver was patched, we need to look at how his score evolved over time. In the simplest case (as in the example above) we expect first low scores followed by scores close to 1. As the patch might have been applied in the middle of a day, this day may get an average score. Score values fluctuate, especially if we see only a modest number of queries from a resolver. Any classification algorithm thus cannot draw too much conclusions from a single score value.

We thus use the following averages as input into the classification:

- "startavg"

  This is an average strongly tilted to the first recorded score values.

- "endavg"

  This is the reverse: the more recent score values much higher weight than earlier scores.

Based on these two numbers, we categorize resolvers into (first match counts):

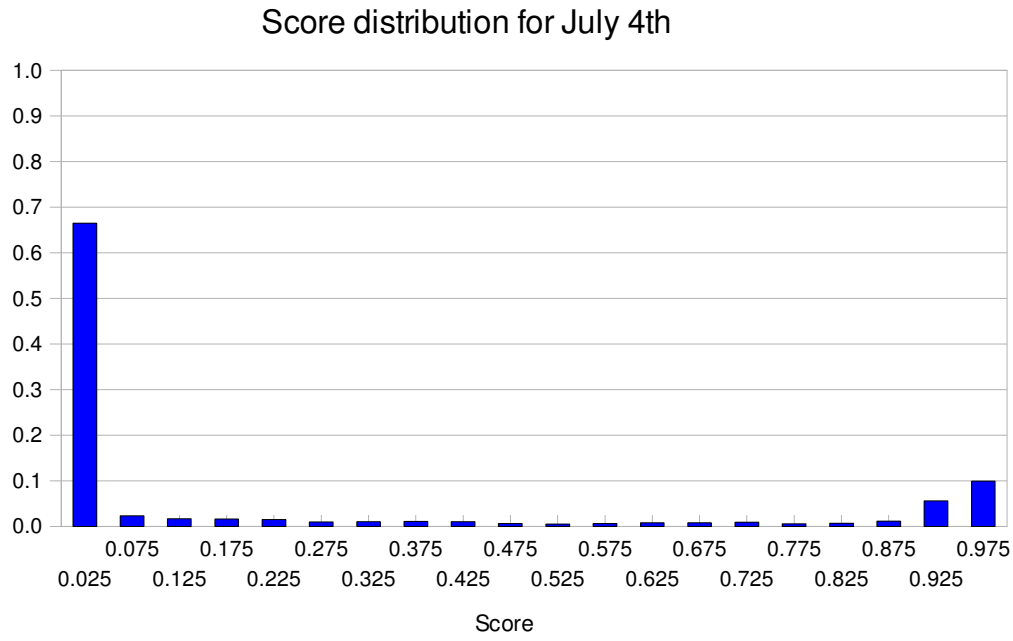
startavg	**endavg**	**Classification**
< 0.2	< 0.2	UNPATCHED
< 0.2	> 0.8	PATCHED
< 0.2	> 0.5	IMPROVED
> 0.2	< 0.8	MEDIOCRE
> 0.8	> 0.8	GOOD
		UNKOWN

Additionally, we try to detect the day when a resolver was patched based on a jump of at least 0.5 in the score rating.

Results

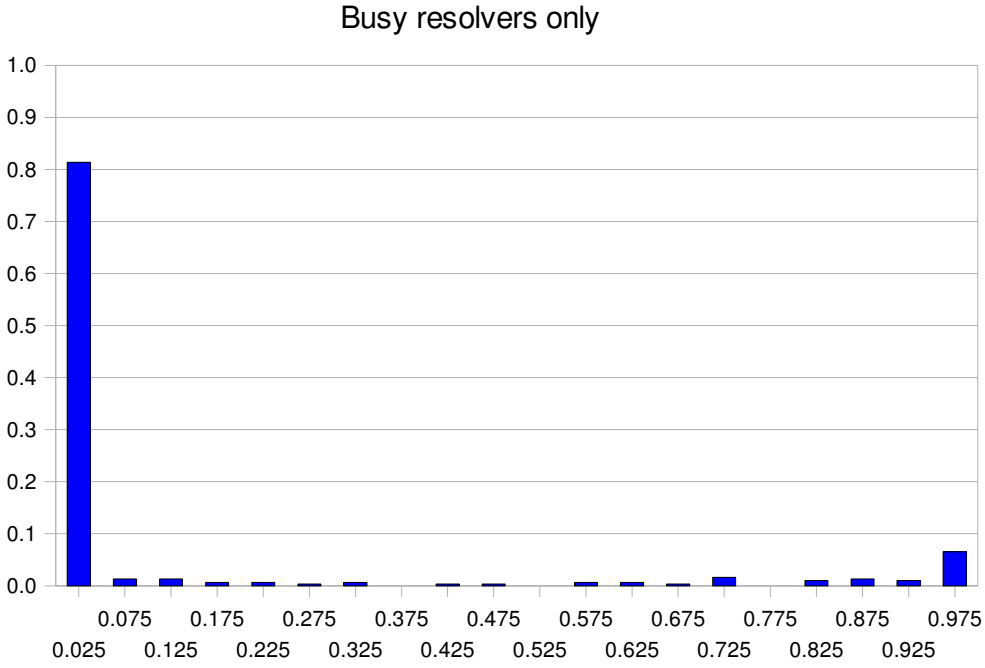
Before the announcement

This chart covers all resolvers from which we saw more than 10 queries on July 4th.



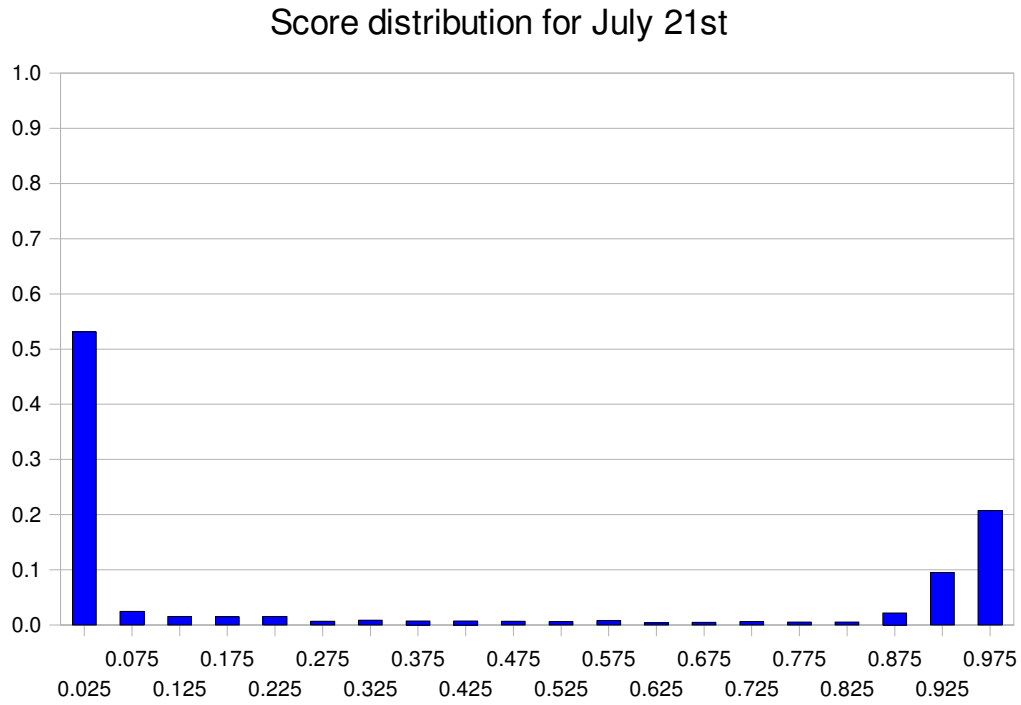
A large majority of resolvers fall into the bin with the lowest scores. Only about 15% already seem to implement SPR. These are probably powerdns or djbdns installations.

The diagram doesn't change much when looking only at the 306 resolvers which sent more than 1000 requests to our nameserver:

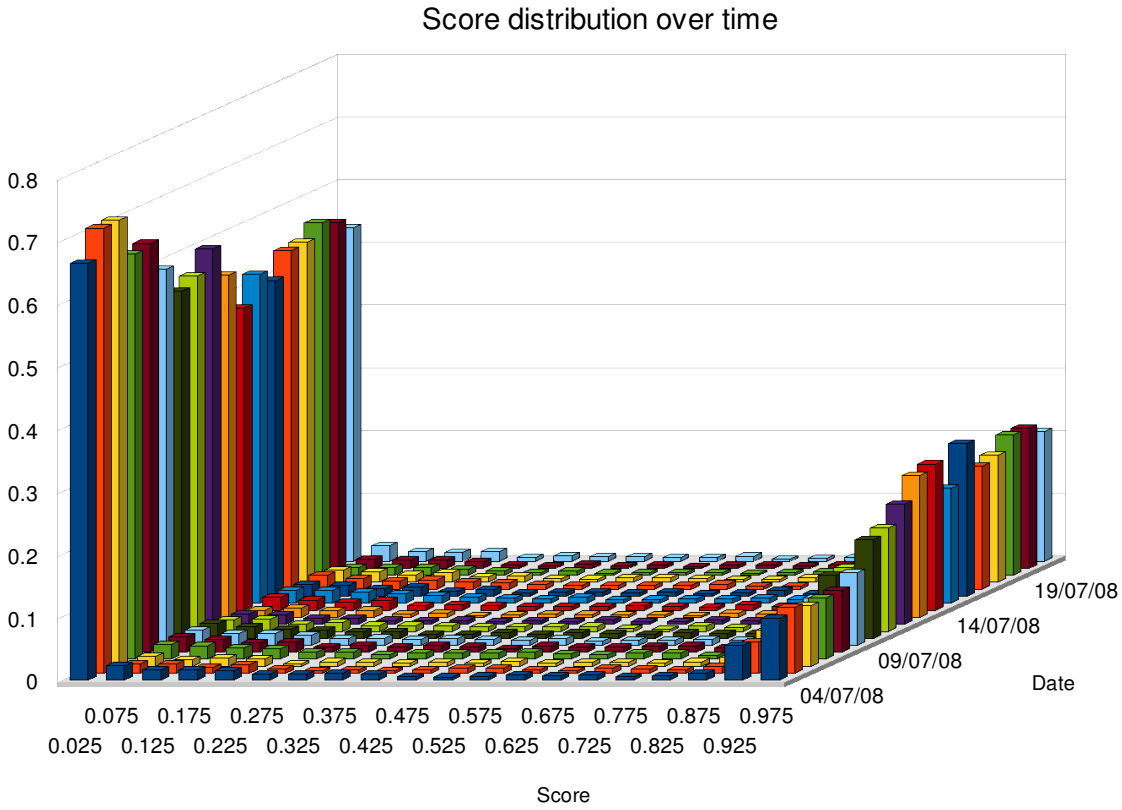


Two weeks later

Almost two weeks later, on July 21st, the chart has improved just slightly. There are still more than 50% unpatched resolvers in Austria.



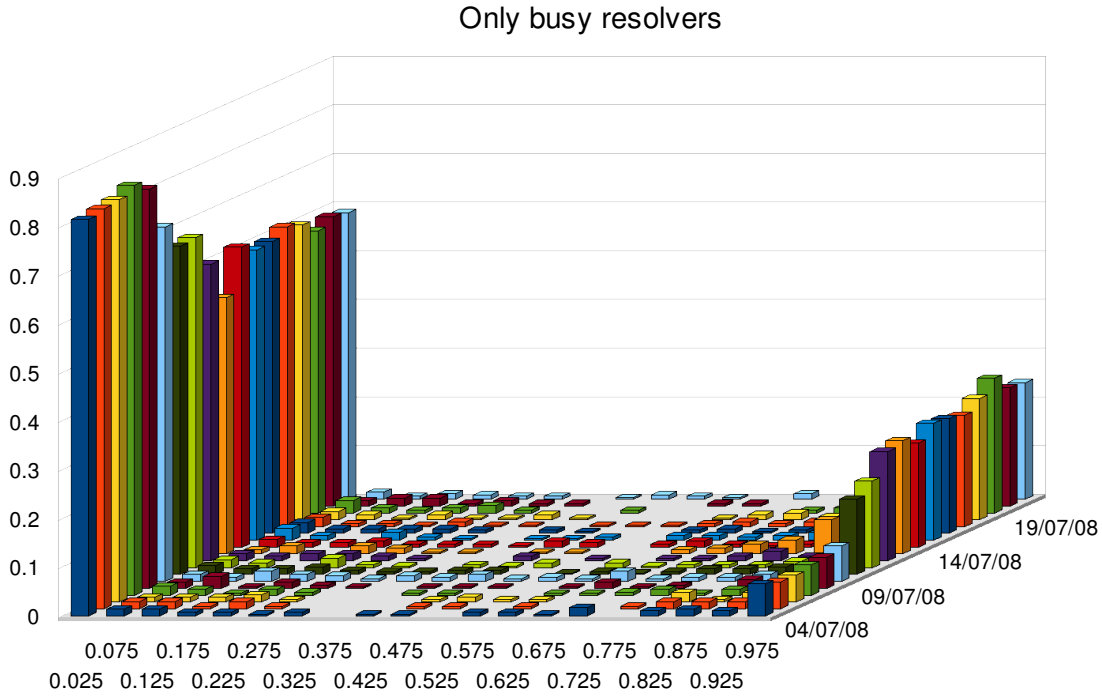
Putting these histograms for all days together into a 3D chart, we can see the changes over time:



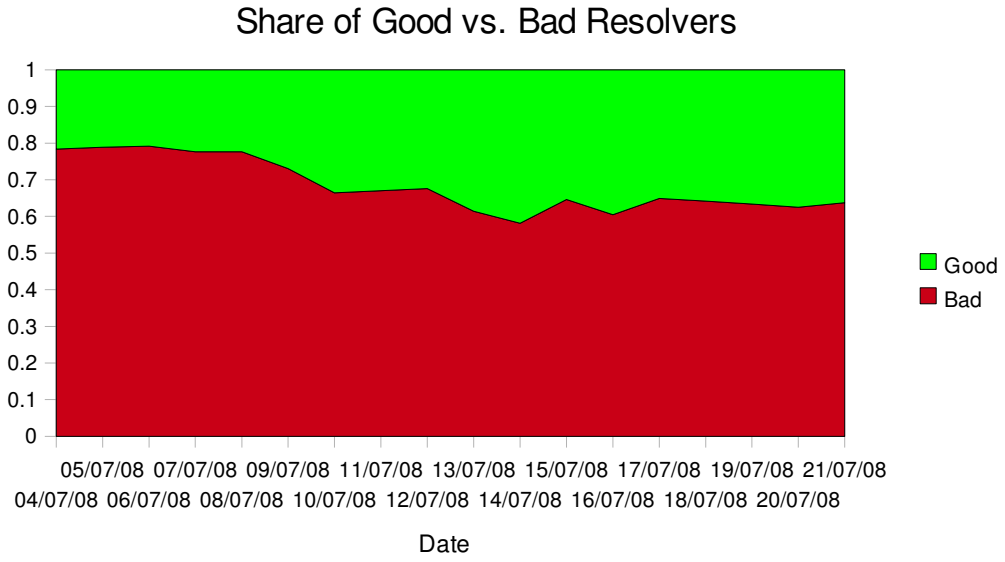
One might think that the larger resolvers (i.e. the ones which send more queries) are better

maintained. This is true to a degree: one can clearly see the few who patched within two days, but the rest does not seem to care.

The good news here is that a manual look into the data shows that some of the really large ISPs in Austria have already patched their servers.

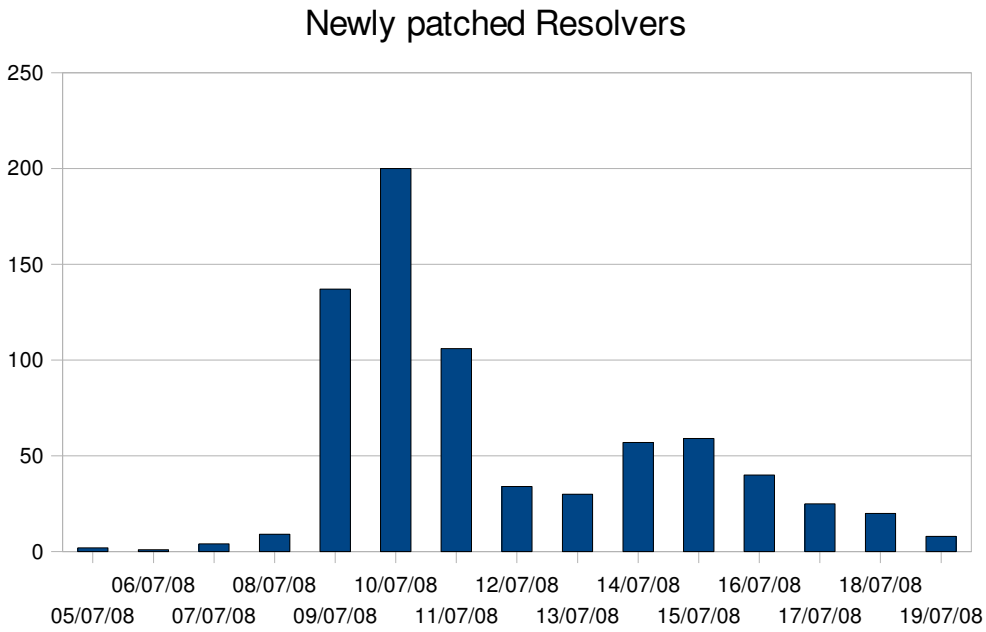


Grouping resolvers into “Good” and “Bad” ones based on score > .5 and score < 0.5 we get the following chart (covering all resolvers):



When did people patch?

As mentioned above, we tried to detect when a specific resolver started to do SPR.

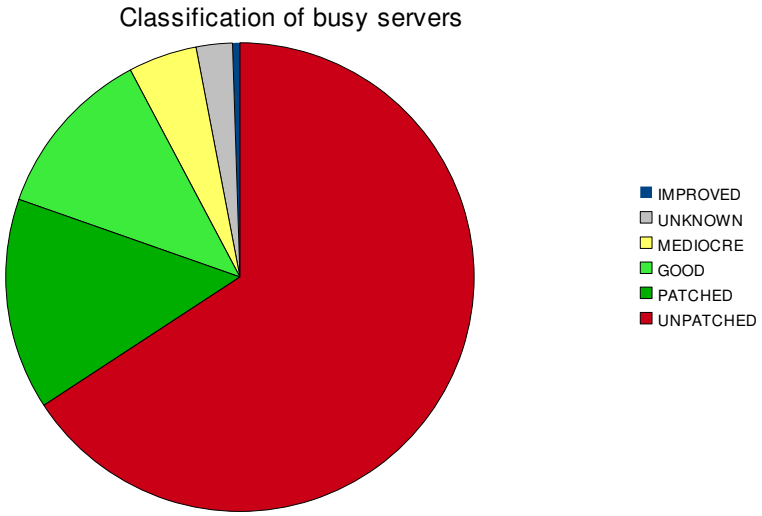


As expected, a good number of people upgraded their software soon after the announcement. The weekend 12th/13th saw a drop-off in the number of newly patched servers.

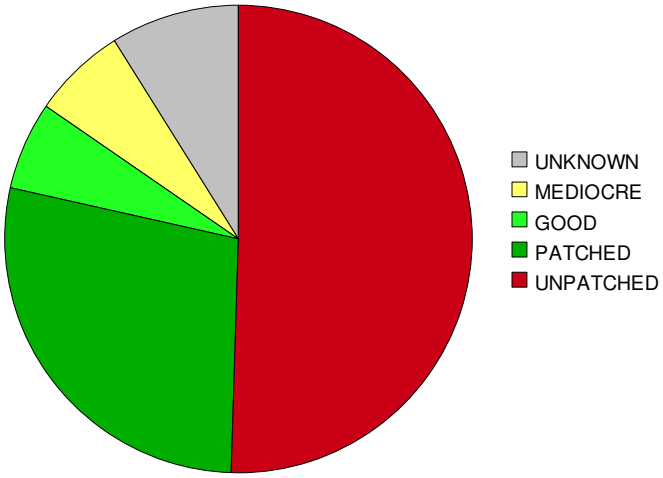
Classification

Our datasets covers 18 days, four of which were weekend. Looking at all servers which appear in at least 14 days, we find the following classification:

Roughly speaking, one quarter of all relevant resolvers implement source port randomization now, whereas the vast majority does not.



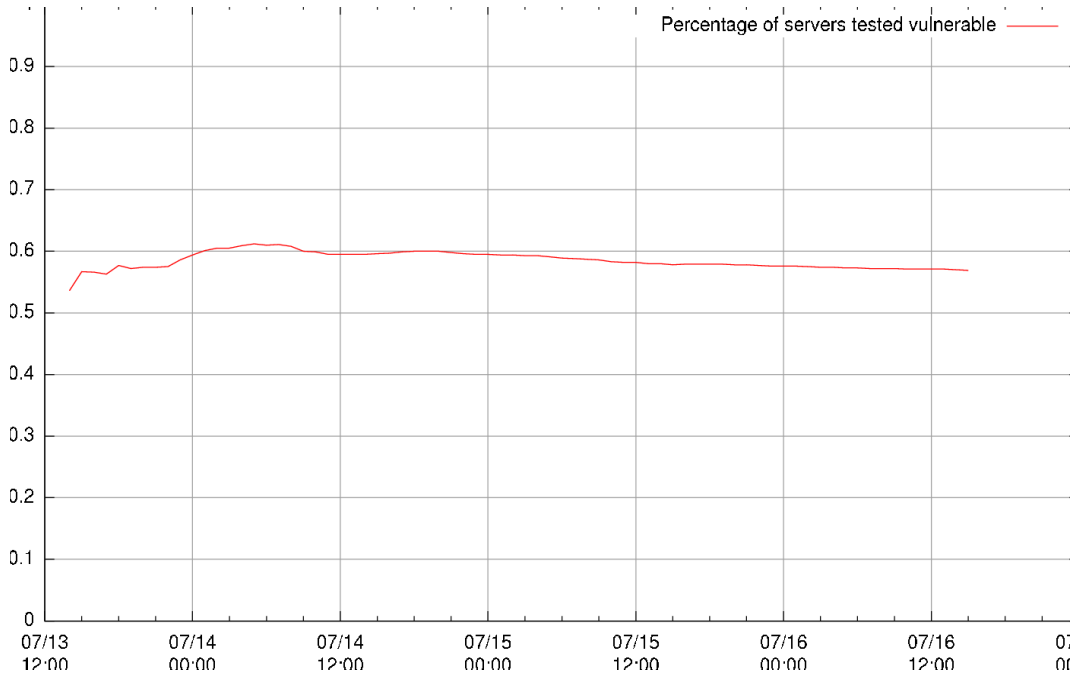
Weighting resolvers based on the number of queries shows a slightly better picture:



About half of all DNS queries are made by unpatched resolvers and thus are vulnerable to DNS cache poisoning.

Comparison with other research

Alexander Klink presented his findings[7] on the patch rate based on the data from Dan Kaminsky's doxpara online testing site. Klink was able to assess the total amount of patched systems vs. unpatched systems based on the queries which went to doxpara.com. As you can see the very slow rate of patch adoption matches our findings.



Outlook

Grim.

Now that exploit code has been published[8], we have to expect successful attacks exploiting this vulnerability.

Acknowledgments

Thanks go to the nic.at Server operations team for the prompt help with the query logs.

7 http://www.shiftordie.de/articles/Traversing%20Dan%27s%20directory
8 http://www.caughq.org/exploits/CAU-EX-2008-0002.txt