HARD-DNS: Highly-Available Redundantly-Distributed DNS

Carlos Gutierrez, Rajesh Krishnan, Ravi Sundaram and Fangfei Zhou

***Abstract*—**

The DNS or Domain Name System is a critical piece of the Internet infrastructure. In recent times there have been numerous attacks on DNS, the Kaminsky attack being one of the more insidious ones. Current solutions to the problem involve patching the DNS software (Bind) and/or using DNSSEC. Unfortunately, these are forklift upgrades of the DNS infrastructure and are not always feasible especially in sensitive and critical installations.

We propose and develop the architecture for HARD-DNS - a turn-key bolt-on solution with no client-side change. We utilize a separate distributed network, HARD-DNS, which is architected for greater resilience to DDoS (Distributed Denial of Service) attacks. We employ quorum techniques to increase tolerance to cache poisoning and we protect the connection between the resolvers and HARD-DNS by a technique we call IP-cloaking. We present theoretical analysis and experimental evaluation to validate the feasibility of our approach.

***Index Terms*— DNS (Domain Name System), Security, DoS (Denial of Service), Internet, Architecture, CDN (Content Delivery Network).**

I. Motivation

"Your browser wouldn't be able to go anywhere; you wouldn't be able to send e-mail. Nothing on the Internet would work"

--Prof. Tom Leighton, Founder & Chief Scientist of Akamai Technologies Inc., on what would happen if the Domain Name System (DNS) were to be targeted by a denial of service (DoS) attack.

In July 2008, at the Black Hat Conference held in Las Vegas, the Kaminsky DNS vulnerability was announced. The Domain Name System or DNS [1] is a critical and integral part of the Internet. Kaminsky [2,3,4] showed that DNS caches throughout the world were susceptible to cache poisoning. This could lead to large-scale impersonation attacks by fake websites. Not only could it potentially lead to the personal and financial ruin of individuals but successful poisoning of this system could also result in catastrophic consequences at a national level [4,5] were any infrastructural networks such as the military network domains to be targeted by an enemy. Currently, there is only one way to mitigate this vulnerability, patching the DNS server or patching the DNS protocol itself. Patching the DNS protocol, i.e. substituting DNSSEC [1] instead of DNS requires a forklift upgrade at both the client and resolving name-server ends and for now the world has settled for updating and patching the DNS server software (typically BIND [1]). The problem with the patch however, is that it requires upgrades to all client side resolving name-servers, which is the more vulnerable and numerous end of things. It is also difficult to upgrade resolving name-servers that are in critical operational paths and need to be running 24/7. While this patch reduces the risk of DNS poisoning by an attacker (by using randomization over a large port space), the patch by itself does not eliminate the threat from a formidable enemy who has the ability to bombard the resolving name-server with large quantities of spoofed responses.

Hence it is clear that there is a need for a solution that does not require upgrading the software at the resolving name-server end. This will enable easy adoption of the solution. At the same time the solution must be robust enough to fend off large-scale attempts to poison the cache.

II. Our Contribution

We propose the use of a distributed network, HARD-DNS, to dramatically reduce the probability of suffering from a poisoned cache in the DNS. We propose to leverage HARD-DNS machines as the public resolvers, by massively distributing the resolution functionality. In this fashion, the client's or ISP's name server will never be exposed to the attacker. When a user types in a domain such as *www.foo.com* into their browser, their name server contacts a random HARD-DNS machine which then recursively resolves the name on their behalf and returns the answer to the customer's name server (or resolver software).

The objective of the proposed *Highly-Available Redundantly-Distributed DNS* (HARD-DNS) approach is to develop a robust protection technique that will minimize the chance of suffering from cache-poisoning attacks. HARD-DNS achieves this by:

Carlos Gutierrez (carlos.gutierrez@ssci.com) and Rajesh Krishnan, (rajesh.krishnan@ssci.com) are with Scientific Systems Company, Inc., Woburn, MA USA.

Fangfei Zhou (youyou@ccs.neu.edu) is a PhD student and corresponding author Ravi Sundaram (koods@ccs.neu.edu, phone: 617-373-5876; fax: 617-373-5121) is an Associate Professor in the College of Computer Science, Northeastern University, Boston, MA USA.

- using quorum techniques and performing a majority vote on responses from multiple individual DNS servers within the HARD-DNS network to obtain a reliable response

- using IP-cloaking to hide the requests from the resolving (ISP) name servers to prevent cache poisoning through concerted spoof attacks

- taking advantage of the global and robust placement of name servers in the HARD-DNS network

HARD-DNS can be implemented as a standalone network or built on top of platforms such as Content Delivery Networks (CDNs), or Cloud Infrastructures. On the customer side the scheme involves nothing more than a configuration change to their resolving name-servers (resolvers). HARD-DNS can be built out gradually providing additional robustness with each new server. Existing CDNs (such as Akamai or Limelight Networks) or other Cloud Computing Infrastructure (such as Amazon's EC2 or Microsoft's Azure) can also be repurposed as HARD-DNS. Our solution provides greater security in incremental fashion at low upgrade cost. Here are some of the primary benefits of HARD-DNS:

- it requires no software upgrade or patch; with minimal configuration changes on the (client) side of the resolving name-server, it can easily be rolled out to the entire base of vulnerable DNS systems

- it is a low-cost incremental approach that can be bootstrapped from an initial network of only a few machines. Of course, the more machines that are part of the HARD-DNS distributed network the more the robustness of the solution against DDoS attacks

- it increases performance and reduces congestion on the Internet ultimately enhancing end-user experience

- it provides security both against known attacks as well as zero-day attacks since the task of resolving is essentially outsourced to the HARD-DNS network.

These benefits could come at the cost of a performance penalty since client side name-servers no longer cache resolutions. However, as we show in the experimental evaluation section (Section VI) of this paper, a well-distributed HARD-DNS network incurs minimal overheads.
The rest of the paper is organized as follows: in Section III we discuss related work. Section IV explains the basic working of DNS as well as the Kaminsky attack which is a prime representative of the class of attacks that achieves cache poisoning though clever use of a denial of service. Section V contains a detailed description of our HARD-DNS solution along with a theoretical analysis of its effectiveness. We present performance results for a PlanetLab [24] deployment of HARD-DNS and conclude in Section VII.

III. Related Work

DNS [1] was not designed with security in mind. To address this shortcoming, Domain Name System Security Extensions (DNSSEC) [1] was developed to provide security for certain kinds of DNS information, such as origin authentication and data integrity but, as mentioned before, its implementation requires a forklift upgrade. DNSSEC is currently being deployed by the US Federal Government with the aim of having all Federal .gov domains and sub-domains configured to use DNSSEC by December 2009 [5]. Other agencies of the US Government, such as the Department of Defense (DoD) are also in the process of deploying DNSSEC, but certain types of networks (e.g. tactical networks), do not have the required infrastructure in place and require alternative means of protection against DNS cache-poisoning in the interim.

It is not clear whether DNSSEC will ever fully supplant DNS. In the meantime the Internet continues to be susceptible to a variety of DDoS and cache poisoning attacks [6]. The Kaminsky attack, a cache poisoning attack, has been detailed extensively [2,3,4]. Techniques for preventing DDoS [14] attacks using CDNs [6,8,9] and Cloud Infrastructures [10,11] have been studied but these are in the context of content delivery and not DNS. A few schemes have been proposed to improve resilience in DNS. CoDNS [21] masks delays in legacy DNS by diverting queries to other healthy resolvers. Schemes [19, 22, 23] built on top of peer-to-peer networks have also been proposed to enhance fault-tolerance and load-balancing properties. Game-theoretic approaches tailored for DNS-specific attacks also exist in the literature [7,15,16].

To the best of our knowledge, this work is the first to propose a solution to the cache poisoning problem by employing a separately hardened distributed network for name resolution in conjunction with the use of quorums and IP-cloaking.

IV. Background

Since HARD-DNS protects against DDoS and cache-poisoning attacks against DNS of which the Kaminsky attack is particularly well-known, we will give a brief background on DNS as well as the Kaminsky attack using illustrative examples. [1] is a definitive source on DNS and [2,3,4] are good descriptions of the Kaminsky attack. For purposes of illustrations we will work with .mil domains in our examples.

A. The Domain Name System (DNS)

The Domain Name System (DNS) is a distributed hierarchical naming system for computers, services, or any Internet resource. Most importantly, it translates domain names meaningful to humans into the numerical (binary) identifiers associated with networking equipment for the purpose of locating and addressing these devices world-wide. An often used analogy to explain the Domain Name System is that it serves as the "phone book" for the Internet by translating human-friendly computer hostnames into IP addresses. For example, www.example.com translates to 208.77.188.166.

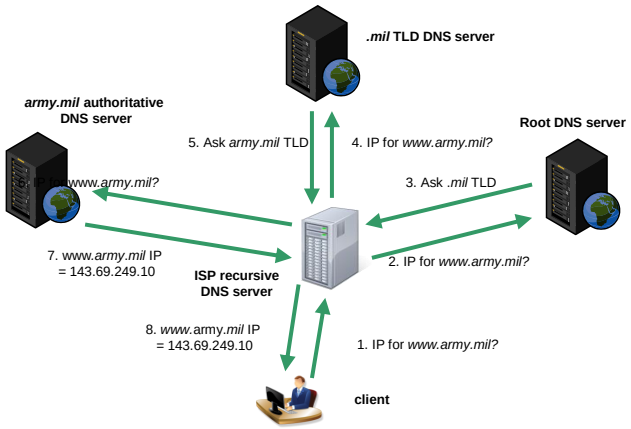

Figure 1: Domain name resolution using DNS

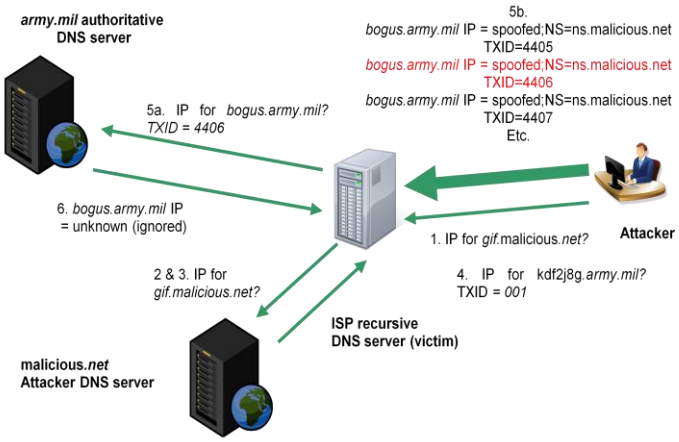

Figure 2: Flow of the Kaminsky Attack

The resolution of a domain address can be seen pictured in Figure 1. A client, attempts to visit a web page, for instance *www.army.mil* and needs to know the web pages IP address. If it does not already have it, the client (web browser) contacts the recursive DNS resolver belonging to its ISP. This resolver will then contact one of the DNS root servers. The root server does not contain the IP address, but it knows who the authoritative DNS name server for the *.mil* zone, or top-level domain (TLD), are and it will send the resolver this information. The resolver will then contact of the *.mil* TLD name servers which will reply with the authoritative name servers for the *army.mil* domain. The *army.mil* authoritative name server does know the IP address for *www.army.mil* and will send *IP = 143.69.249.10* as the reply to the ISP name server. Finally, the ISP name server will send *IP = 143.69.249.10* to the requesting client. To prevent unnecessary queries from being sent, the ISP name server can cache previously received replies and use those during the resolution process if necessary. For instance, if the client were to request the IP address for *www2.army.mil*, the resolver can now contact the *army.mil* authoritative DNS server directly.

B. DNS Cache Poisoning – The Kaminsky Attack

In summer of 2008, Dan Kaminsky [1] announced his discovery of a serious vulnerability in DNS aimed at recursive name servers [2][4]. It relies on a previous vulnerability called 'CNiping' by which a name server will update its records when it receives a record containing a CNAME (or alias) update even if that record is not expired. In Figure 2, the information in the answer section will be overwritten by the new reply even if it was not expired. Thus, a malicious user can spoof a DNS packet and update this information.

The basic idea of the Kaminsky attack is to spoof answer packets from the authoritative name server to the requesting name server for a name in the authentic domain. The victim name server is provoked to go to a name server run by the attacker by either using it to recursively resolve a name in the attacker's domain or by putting a dummy pixel on a webpage belonging to the attacker's domain. When the victim server queries the attacker's name server the attacker learns the victim's IP address as well as source port. The attacker then provokes the name server to request a fake name in a real domain. Now the attacker sends a flood of spoofed response packets to the victim name server in an attempt to get a packet in before the genuine answer from the authoritative name server. The reason a flood of packets has to be sent is that the attacker has to get the TXID, a randomly chosen two-byte field, correct. If the attacker can get its packet in before the real packet then it can not only change existing cache entries corresponding to genuine names (CNiping) it *can even change the IP addresses for authoritative name servers for the domain*! In this way all clients of the victim name server will end up going to the impostor web sites set up by the attacker. The Kaminsky attack can be used to hijack potentially even the root servers and therefore the entire root domain. To increase the chances of this attack succeeding, a bandwidth attack, or Denial-of-Service (DoS) attack can be performed in parallel against the authoritative name servers. This will slow down the reply from the authoritative name servers to the victim name server by increasing the time window during which it can send the forged packets. Figure 2 shows a diagram of the Kaminski attack. The attacker begins by requesting the IP address for a bogus host from the victim name server. The name server will then ask the authoritative name server for the IP address (step 5a). At the same time, however, the attacker floods the victim name server with forged packets that will answer its question to the *army.mil* name server, each with a guessed TXID. If one of the forged packets contains the correct TXID (4406) of the victim's request to the *army.mil* name server (in red in 5b), the victim name server will accept it and update its records. This forged packet not only contains the IP address for the bogus host, but

it will also contain information that points the *army.mil* name servers to a host controlled by the attacker, as shown in Figure 3. Finally, since the reply from the authoritative name server arrives too late, it will automatically be discarded.

```
;; QUESTION SECTION:
;kdf2j8g.army.mil.                IN      A

;; ANSWER SECTION:
kdf2j8g.army.mil.           274         IN          A
A.B.C.D

;; AUTHORITY SECTION:
us.army.mil.                 600        IN         NS
ns.attacker.net.

;; ADDITIONAL SECTION:
ns.attacker.net.             69712      IN          A
A.B.C.N
```

Figure 3: Forged DNS reply of the Kaminsky attack

V. The HARD-DNS Approach

A. Overview

The basic idea is to solve the DDoS and cache-poisoning attacks by outsourcing the task of resolving name lookups to a fault-tolerant distributed network, HARD-DNS. We propose to leverage HARD-DNS machines as the public resolvers, by massively distributing the resolution functionality. This will require no software or hardware change whatsoever on the side of the resolving name servers. Name servers are configured with a hint zone consisting of the names and IP addresses of the root servers. We propose to replace this with the names and IP addresses of HARD-DNS nodes. Thus the name server contacts these CDN nodes to get its resolutions (see 4). The HARD-DNS nodes are set up to query recursively and then combine their answers using majority to diffuse the effect of poisoning. Observe that the original name server is completely shielded from the Internet because it never directly queries any other name server. HARD-DNS nodes resolve their queries by contacting the appropriate authoritative name servers.

If the attacker were to attack or attempt to cache-poison they would instead attack a particular HARD-DNS machine. This would, at most, poison the cache of that particular HARD-DNS machine and since the customer picks a random HARD-DNS machine each time, they would be safe the overwhelming majority of the time. This will render the cache-poisoning attacks ineffective. The HARD-DNS network uses overlay routing to distribute the answers amongst themselves and arrive at quorums and consensus. Overlay routing confers superior benefits of speed and reliability over native BGP routing. To compute its response HARD-DNS polls a subset of its name servers and then returns the majority vote. This would fail only if the attackers are able to poison the caches of all but a small fraction of the HARD-DNS machines, a virtual impossibility since HARD-DNS can be architected to have tens of thousands of machines. HARD-DNS also utilizes load-balancing techniques to diffuse the effect of DDoS attacks and IP-Cloaking techniques to hide the requests from the resolving name-server to the HARD-DNS network. IP-cloaking provides greater security than port randomization which is the prevalent technique for mitigating the Kaminsky attack.

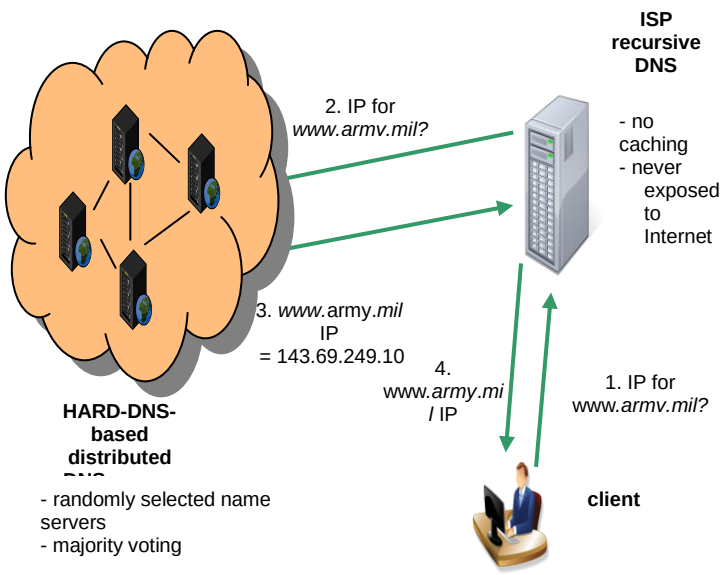

Figure 4: Robust DNS infrastructure using HARD-DNS.

B. BGP Anycast and Overlay Routing

The requests from the (client) resolving name server are directed to the optimal HARD-DNS server using BGP anycast [17]. The entire HARD-DNS network also functions as an overlay network and content is moved through the network from one end to another via intermediate relays. Even though each hop between CDN nodes is dictated by BGP (Border Gateway Protocol) nevertheless overlay routing is often superior to the native BGP routing between the same endpoints. This is because BGP is not aware of the congestion in the network [18].

C. Load-balancing and Attack-diffusion

Load balancing amiong the HARD-DNS servers can be achieved using one or more layer 4–7 switches to share traffic among a number of DNS servers. This has the advantages of balancing load, increasing total capacity, improving scalability, and providing increased reliability by redistributing the load of a failed DNS server and providing server health checks.

The main advantage of a HARD-DNS network is that the entire process of resolution is offloaded from the (client) resolving name server) thus shielding the location, even the existence, of the resolving (client) name server from the public Internet. Users and attackers only see the HARD-DNS nodes. Attackers can choose to attack HARD-DNS nodes but a well-provisioned network will diffuse the attack through load-balancing. When one of the nodes comes under attack HARD-

DNS automatically does load balancing to move the load away to other servers. If the attacker chooses to move to the new servers then automatically the old servers spring back to life and start serving traffic. If the attackers stay with the original nodes then new traffic automatically gets served from other nodes. In this way, HARD-DNS protects the origin website against denial of service and degradation. The general mechanism of protection can be described by an analogy – just as important dignitaries are protected by big bodyguards that effectively act as bullet catchers so too does HARD-DNS protect the origin by absorbing the attack through its numerous and well-provisioned server clusters that distribute the load and diffuse the onslaught.

D. IP-Cloaking

The common approach to dealing with the Kaminsky approach is to mitigate the chance of a spoofed packet having the right TXID by increasing the space of possibilities using port randomization. The resolving name server uses a random port from a space of a few thousand ports and since the spoofed packet has to get both the port and the TXID right to be able to poison the cache, this greatly increases the level of security. We extend this idea significantly using the concept of IP-cloaking. The HARD-DNS nodes are IPed with a large IP space. The resolving name-servers are then set up to employ hashing with a shared secret key to determine the specific IP address at the given time to contact. The HARD-DNS servers will answer only to the specific key at the specific time. The hashing can be done using a simple scripting language to modify the configuration file and does not require any software or hardware upgrade. The hint zone of root servers is periodically updated with the list derived from this shared key and soft-uploaded to the name server (name servers are already set up to soft-upload new configuration files) so that the original name servers contact the appropriate IP addresses. This provides an additional layer of protection because even if the IP addresses of the origin name servers are exposed nevertheless the attacker must correctly guess the HARD-DNS IP addresses derived from the shared key to be able to spoof their response – an impossible task for a sufficiently large IP space with which the HARD-DNS nodes are IPed (e.g. a Class B address space with 64000 IP addresses).

E. Quorum technique for fault-tolerance

We use a quorum based approach to reduce the chance of cache poisoning. For each resolution request from a (client) resolving name server, HARD-DNS utilizes a number of its servers, each querying for the resolution and then takes the majority response from them. Suppose we say that 2 *k* HARD-DNS nodes query for the requested resolution and the majority response is returned back to the original name server. This response can be poisoned only if at least *k*+1 of the HARD-DNS nodes got poisoned. Studies of the Kaminsky attack show that it succeeds in about 10 seconds in the most optimistic case. If we assume that a server gets poisoned with probability 1/10 in a given second then the probability that at least *k*+1 of them get poisoned is at most

$$2k\text{-choose-}(k+1) * (1/10)^{(k+1)} <= 10^{-(0.8k)}$$

Thus if we choose k =50 we get that the probability that the majority response is bad is at most 10^{-40}, i.e. expected time for the attack to succeed is 10^{40} seconds which is in excess of the life of the universe.

F. HARD-DNS and CDNs

Observe that HARD-DNS can be easily built on existing distributed platforms such Content Delivery Networks (CDNs) or Cloud Infrastructures. CDNs have an edge over Cloud Infrastructures since they internally use DNS which can be easily modified to support HARD-DNS and they already posses a redundant and robust global footprint. It must be noted that we only rely on the CDN's DNS infrastructure and not on the CDN's content caching infrastructure. Furthermore, HARD-DNS truly requires only a large set of redundant DNS name servers; if this infrastructure can be provided by other means such as through a Cloud Infrastructure provider, a commercial CDN is not a necessary requirement.

VI. Experimental Evaluation

We deployed HARD-DNS on PlanetLab [24] which is an open platform available to the academic community for designing, implementing and testing new large scale services. The advantage of using a platform such as PlanetLab as opposed to just testing in the lab is that we can evaluate HARD-DNS in the context of real world congestion and losses. We set up a HARD-DNS network of 6, 8 and 10 servers on geographically distributed nodes. We used BIND version 8 which is a legacy version with the vulnerability to the Kaminsky attack. We used 1 server by itself as the control case. For a network of 2k + 1 servers we used a random subset of k+1 as the quorum, i.e., with 10 HARD-DNS servers the resolving name-server would contact 6 randomly chosen servers. We were continuously bombarding all servers with the Kaminsky attack. Now of course the duration of a poisoning is a function of the TTLs (Time-To-Live). Since our goal is to see how quickly a given network (control server vs. HARD-DNS) gets poisoned but not how long it stays poisoned we would reset all machines every time we received a poisoned response from either the control server or the HARD-DNS network. We set long TTLs (10S) on all returned responses which means that when a server got poisoned it would normally stay poisoned for a long time (10 seconds) but then we would reset all machines every time we detected a poisoned response, as explained before. We ran requests every 1 second for a resolution from both the control server as well as from HARD-DNS network, for a total of 12 hours. We measured the latency of lookups as also the time to poisoning. Our results should be viewed in the context of our setup on

PlanetLab where we were averaging about 120ms RTTs (Round-Trip Times).

We now present results on the performance and reliability of the HARD-DNS network vs the control server. Figure 5 shows the cumulative distribution of lookup latencies incurred by HARD-DNS vs. the control server while Figure 6 shows the cumulative distribution of the time to poisoning.

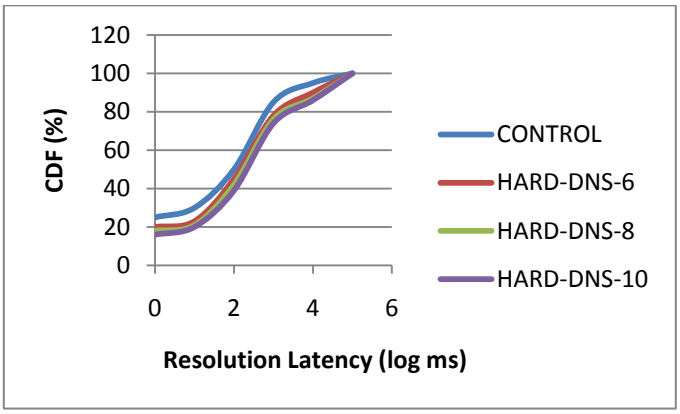

Figure 5: CDF of Latency of Resolution.

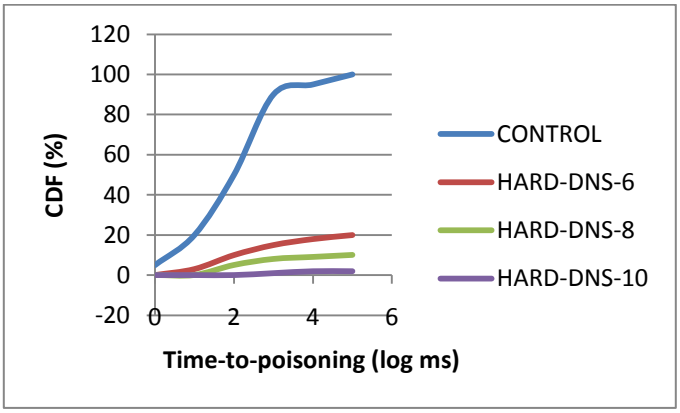

Figure 6: CDF of Time-to-Poisoning.

The main takeaway from these results is that HARD-DNS provides reasonably low latencies while exhibiting great resilience

VII. Conclusion

We have developed the architecture for a solution to the problem of DDoS and cache poisoning attacks against DNS. Our HARD-DNS proposal has the merit that it is a low-cost incremental approach that does not require any software upgrade on the side of the clients, or existing authoritative name servers. We have obtained a provisional patent for our solution [20].

One interesting direction that might be worth exploring is to consider a distributed network where the network is composed of different strategic agents with different utility function, some even potentially malicious. As our solution stands currently we do not consider that the HARD-DNS network may be subject to insider attacks.

Another interesting direction would be to provide additional services through HARD-DNS such as privacy and anonymity or even transcoding services for mobile devices.

References

[1] Liu, C., Albitz, P., *DNS and BIND*, O'Reilly, 2006.

[2] Kaminsky, D. *DoxPara Research*, http://www.doxpara.com

[3] Olney, M., Mullen, P. and Miklavcic, K. *Dan Kaminsky's 2008 DNS Vulnerability*, Sourcefire Vulnerability Report, July 25, 2008, http://www.snort.org/vrt/docs/white_papers/DNS_Vulnerability.pdf.

[4] Friedl, S. *An Illustrated Guide to the Kaminsky DNS Vulnerability*, August 07, 2008, http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

[5] *Securing the Federal Government's Domain Name System Infrastructure*, White House Memoranda, August 22, 2008. http://www.whitehouse.gov/omb/memoranda/fy2008/m08-23.pdf

[6] F. T. Leighton *Internet Security Insight*, Akamai White Paper http://www.akamai.com/html/perspectives/insight_tl_internet_security.html

[7] R. Sundaram, M. Snyder and M. Thakur, *Preprocessing DNS log data for effective data mining*. To appear in Proceedings of ICC, 2009.

[8] R. Sundaram, M. Afergan, A. Ellis and H. Rahul, *Method and system for protecting websites from public Internet threats*," Patent #: 7,260,639. Granted August 21, 2007.

[9] R. Sundaram, H. Rahul, *Method and system for providing on-demand content delivery for an origin server*," Patent #: 7,376,736. May 20, 2008.

[10] Amazon Elastic Compute Cloud (EC2) http://aws.amazon.com/ec2/

[11] Microsoft Windows Azure Platform, http://www.microsoft.com/azure/default.mspx

[12] Botnet attack of 2004, http://news.zdnet.com/2100-1009_22-136640.html

[13] Continuing attacks of 2009, http://www.akamai.com/html/about/press/releases/2009/press_070909.html

[14] Christos Douligeris and Aikaterini Mitrokotsa. *Ddos attacks and defense mechanisms: classification and state-of-the-art*. Comput. Networks, 44(5):643–666, 2004.

[15] Tzi-cker Chiueh Fanglu Guo, Jiawu Chen. *Spoof detection for preventing dos attacks against dns servers*. In 26th IEEE International Conf on Distributed Computing Systems (ICDCS'06), page 37, 2006.

[16] M. Snyder, R. Sundaram and M. Thakur, A Game-Theoretic Framework for Bandwidth Attacks and Statistical Defenses, Proceedings of IEEE LCN, 2007.

[17] Bornstein, C., Canfield, T., Miller, G., Rao, S., and Sundaram, R., *Optimal route-selection in a content delivery network*, Patent #: 7,274,658, September 2007.

[18] Beijnum, I., *BGP*, O'Reilly, 2002.

[19] Ramasubramanian, V., and Sirer, E., *The design and implementation of a next generation name service for the internet*, SIGCOMM, 331-342, 2004.

[20] Sundaram, R., Krishnan, R., and Gutierrez, C., *HARD-DNS: Highly-Available Redundantly Distributed DNS*, Provisional patent.

[21] Park, K., Wang, Z., Pai, V., and Peterson, L., *CoDNS: Masking DNS delays via Cooperative Lookups*, Princeton University Computer Science Technical Report TR-690-04, 2004.

[22] Cox, R., Muthitacharoen, A., and Morris, R., *Serving DNS Using a Peer-to-Peer Lookup Service*. IPTPS 155-165, 2002

[23] Theimer, M., and Jones, M., Overlook: *Scalable Name Service on an Overlay Network*. ICDCS 2002

[24] Bavier, A., Bowman, M., Chun, B., Culler, D., Karlin, S., Muir, S., Peterson, L., Roscoe, T., Spalink, T., and Wawrzoniak, M., *Operating System Support for Planetary-Scale Network Services* NSDI, May 2004