Making DNS Servers Resistant to Cyber Attacks: An Empirical Study on Formal Methods and Performance

Barry S. Fagin, Bradley Klanderman
Director, Academy Center for Cyberspace Research
Department of Computer Science
US Air Force Academy
Colorado Springs, CO 80840 USA
barry.fagin@usafa.edu

Martin C. Carlisle
Information Networking Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
martin.c.carlisle@gmail.com

IRONSIDES is an open-source Domain Name System (DNS) server designed using formal methods to reduce DNS vulnerabilities to cyber attacks. The use of formal methods gives IRONSIDES provable security properties, including the absence of numerous security flaws that plague BIND and Windows DNS. It also raises an empirical question: Does the use of formal methods to generate provably secure code require sacrificing performance?

We present the results of an experimental investigation to answer this question. We compared IRONSIDES to BIND, Windows DNS, and numerous other DNS servers on both Windows and Unix. Our results show IRONSIDES performs quite well compared to other DNS servers, both proprietary and open-source, particularly given the resources expended in its development. This suggests that, at least in the DNS domain, increasing security with formal methods to render them less vulnerable to cyber attacks does not require sacrificing performance.

Keywords—Ada, DNS, open source, cyber attack resistance, formal methods, internet software, performance analysis, SPARK.

I. Introduction

The Domain Name System (DNS) is the internet protocol that transforms hostnames (e.g. cnn.com) into IP addresses (e.g. 151.101.0.73). Originally proposed by Mockapetris in [1], it is a distributed database protocol that uses the internet as a tree structure to manage records containing information about machine names and properties.

Software that implements this protocol is referred to as a *DNS server*. This term can also describe the machine a software DNS server runs on. Servers that are responsible for resolving names in a single *zone* (typically a company, university, or similarly scoped institution) are called *authoritative* servers. If queried about names outside the zone for which they are responsible, authoritative servers reply with a failure message, the equivalent of "I don't know".

Servers capable of resolving names for any publicly visible machine on the internet are called *recursive.* They use a recursive process to travel up the distributed internet tree structure to determine the name of the machine in question. In modern DNS practice, most recursive solvers do not use full recursion to traverse the name tree. Instead, they refer queries to a publicly available fully recursive DNS server (for example, Google's public DNS at 8.8.8.8), and then cache the result for future use.

DNS is a vital internet protocol. Unfortunately, because it dates from the early days of networking, it contains security flaws that require mitigation to prevent malicious actors from exploiting the system [2]. Additionally, most DNS software is written in older languages with inherent security problems. These languages do not lend themselves to rigorous software design and provable security properties. The two most popular DNS servers, BIND and WINDNS, have a large number of known security flaws that render them vulnerable to cyber attacks, including crashing in response to the injection of bad data and bugs that permit remote execution [3], [4]. These are described in more detail in the sections that follow.

II. Previous Work

In an attempt to address the problem of security flaws in widely deployed internet software, we developed IRONSIDES, a DNS server written in Ada/SPARK [5]. IRONSIDES has numerous provable security properties, including an absence of remote execution vulnerabilities and invulnerability to termination as a result of unexpected or incorrectly formatted input. It runs under both Windows and Linux. More details about IRONSIDES are available in [6] and [7].

Our previous work with IRONSIDES showed that a more secure DNS server can be implemented using formal methods with relatively low development costs (the work was performed by two faculty members at a teaching university working on a part-time research grant). As expected, we were not able to offer the full functionality of an internet consortium (BIND) or a multibillion dollar software company (WINDNS), but we were able to deliver a working authoritative DNS server with provably better security than either [6],[7]. This suggested that formal methods can and should be used to improve the security of both open- and closed-source internet software.

The successful development of IRONSIDES raised two follow-on questions: 1) Can these successes be duplicated in a recursive server, and 2) How does the use of formal methods impact performance? This paper reports our experimental investigation of these questions.

III. Testing Environment

Benchmarking recursive DNS servers is a known hard problem. Network traffic, topology, cache management, security configurations, operating system choice and system administration policies all affect performance. The complex nature of the name resolution task makes it difficult to control for individual variables, to test systems fairly, and to generate repeatable results. There are also a number of benchmarking tools to choose from (for example [8] and [9]), each of which could affect the results produced.

We have attempted to mitigate these effects by testing all servers on a closed network of virtual machines. One VM generates DNS queries, another simulates a connection to the internet, and the rest each run a separate server/OS combination (either Windows or Ubuntu Linux). We describe this in detail below.

A. Hardware and Software Tools

The underlying hardware for our experiment is an NCS Technologies server with a Xenon E5 CPU, 256G of RAM and a 3.7TB RAID hard drive, running XenCenter 7 for instantiating virtual machines. Each VM has 2G of RAM and 16G of disk space. The server can in turn be accessed from workstations in the laboratories of the Academy Center for Cyberspace Research.

All this is abstracted from the experimenter during testing. From the experimenter's point of view, the test bed is best viewed as a collection of virtual machines running various tools, as shown in Fig. 1:

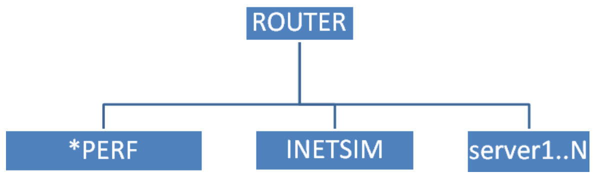

Fig. 1. Experimental test environment

The virtual machine labeled *PERF runs the benchmarking tools. We chose Nominum's DNSPerf and ResPerf [10] due to their open source nature and free availability to the general public. (DNSPerf is used for authoritative servers, ResPerf for recursive servers). We were confident we could quickly get these tools up and running in our environment, and wanted to be able to inspect the source code in the event of problems or the desire to customize them to meet our needs. For the results shown here, customization was not necessary, the tools were built unaltered.

INETSIM is an open-source software suite designed to simulate common internet services in a lab environment [11]. It is used when the behavior of a piece of software is of interest but a direct connection to the internet is either unnecessary or undesirable. This could include malware testing, for example, or in our case testing DNS servers. Using INETSIM enabled us to eliminate external network traffic and name resolution issues from our experiment, focusing exclusively on name server performance. One of the VM's in the test environment is dedicated to running INETSIM. This machine must be up and running when recursive servers are under test.

The VM labeled ROUTER is simply the connection to the external internet, used for downloading software. It does not need to be up during testing if no external resources are required.

B. DNS servers tested

We tested a total of eight DNS servers, briefly summarized below:

1) BIND: Open-source and the industry standard, BIND offers a full complement of DNS functions. Although hard numbers are difficult to come by, in 2010 one online computer security blog estimated that 85% of DNS servers run BIND code [12]. As of three years ago, BIND's site claimed a market share of "over 80%" [13].

Unfortunately, BIND is rife with security problems. As of this writing, there are 78 known security vulnerabilities affecting the current BIND distribution [3]. The provable absence of most of these problems in IRONSIDES is described in [6] and [7].

2) DJBDNS: An open-source DNS server developed by Daniel J. Bernstein. Designed specificially with security in mind, Dr Bernstein offers $1000 to the first person to publicly report a verifiable security hole in his code [14].

3) DNSMASQ: The default DNS infrastructure for open-source Unix systems.

4) IRONSIDES: Implemented in Ada/SPARK with formally provable security properties. Described previously.

5) KNOT_DNS: An open-source, authoritative-only DNS server, developed by the Czech Network Internet Consortium [15].

6) MARADNS: A small, open-source DNS server developed by Sam Trenholme, designed to be lightweight, easy to set up, and very secure [16].

7) POWERDNS: An open-source DNS server provided by a commercial company [17].

8) WINDNS: Microsoft's DNS server that comes bundled with Windows. A longstanding industry standard like BIND, it is also known to have numerous security vulnerabilities [4] provably absent from IRONSIDES.

Of all the servers listed above, only IRONSIDES has formally proven security properties.

C. Experimental Protocol

Testing authoritative servers is fairly simple. All servers are configured with a zone file containing various DNS record entries for machines in the mock zone "dfcs.usafa.edu". DNSPerf is then activated on the benchmarking VM, and run under various shell scripts to send queries at varying rates to the server under test. The query is issued to the target server, the server responds, and the information is logged. At the end of the run, data is dumped for performance analysis and processed with further shell scripts. This is shown in Fig. 2.

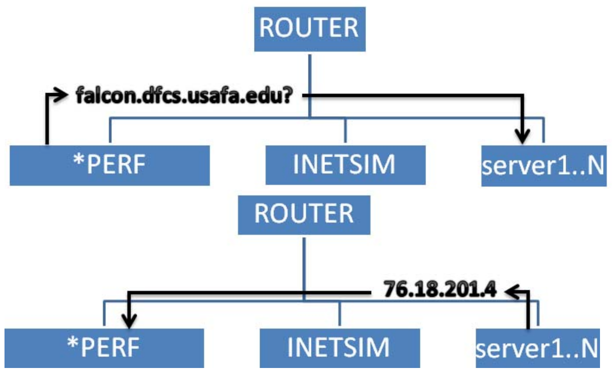

Fig. 2. Authoritative server testing

Testing recursive servers is more complicated. Resperf is activated on the benchmarking machine and pointed to the target. First, it sends a query to the machine running the server under test. The server then forwards the query to the machine running INETSIM (which includes a DNS server simulator). This simulates the forwarding of a query to a publicly available DNS server, something many DNS servers now do.

INETSIM returns a mock address (the same one for all queries) to the resolver under test. That information is cached, and then forwarded to the machine running Resperf. Further queries for that same host name will be handled exclusively by the target as long as the entry remains in the cache. This process is shown in Fig. 3:

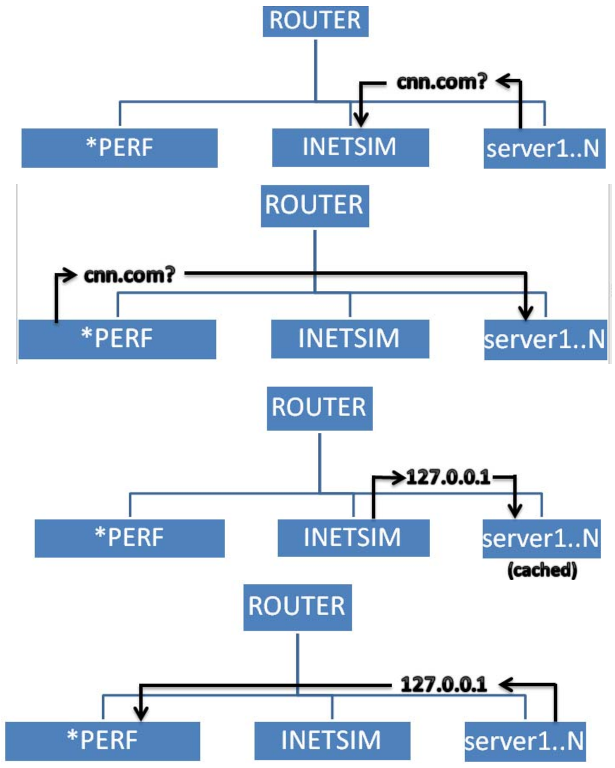

Fig. 3. Recursive server testing

IV. Experimental Results

DNS server performance analysis is hard because there are many different metrics by which DNS performance can be measured. This is further complicated by individual server query management policies. For example, some servers can choose to simply ignore queries when they become too busy, while others may attempt to handle all queries they receive to the maximum extent possible given the limits of their internal data structures.

To maximize the repeatability of these results and fairness of cross-server comparisons, we explicitly define our performance metrics as follows:

Maximum queries per second (qps): For authoritative servers, this is the maximum number of queries that DNSperf reports that the server under test was able to sustain. For recursive servers, this is the command line argument to Resperf, and appears as an x-axis label in the figures below. It is the maximum number of queries per second that Resperf attempts to send to a server during a particular test run.

Latency: The time between a query initiation and response as reported by DNSperf or Resperf.

Queries sent: The total number of queries the tool was able to send to the server during a test run.

% Queries lost: The percentage of queries that were either unanswered or were answered with SERVFAIL. We count failures as lost queries because there is no reason in our test environment to fail other than the inability of the server to handle the load.

Successful queries per second: The total number of queries sent minus the number of lost queries, divided by the length of the test run.

A. Authoritative server performance

Because DNSperf is intended solely for authoritative servers, it only reports the maximum number of queries per second it was able to achieve.

Figs. 4 and 5 show the maximum qps values we obtained for authoritative servers under Unix and Windows respectively.

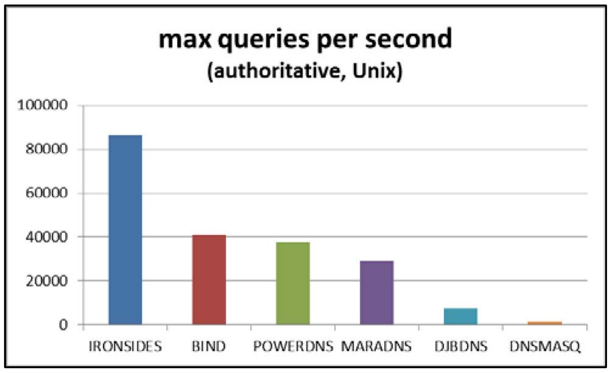

Fig. 4. Max queries per second for authoritative servers running Unix

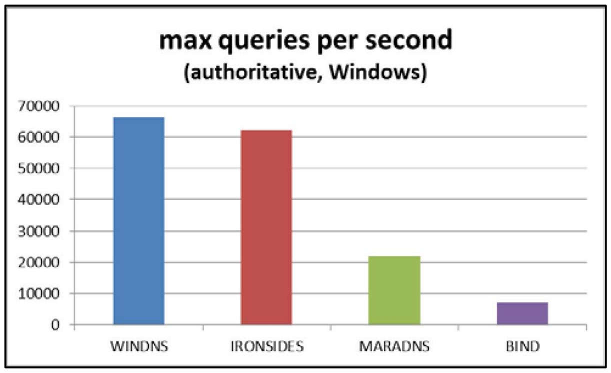

Fig. 5. Max queries per second for recursive servers running Unix

For Unix, IRONSIDES exhibits significantly higher max qps than any others we tested, more than twice that of BIND, its next closest competitor. For Windows, WINDNS performs better, but only slightly (7%).

We would expect WINDNS to perform better under Windows due to its developers' access to the OS kernel and its associated tight integration with the host system. That said, we believe the relatively small difference in performance compared to IRONSIDES is significant, particularly when comparing the resources of a multibillion dollar software company to an academic research laboratory at an undergraduate institition.

B. Recursive server performance

Recursive servers were tested with Resperf with successively increasing values of maximum qps on the command line. As these values increase, server load becomes heavier, more queries are dropped, and latency declines or flattens out. The maximum qps for a given run appears as an x-axis label, with the metric reported on the y-axis.

1) Queries Sent: Fig. 6 shows the total number of queries sent as a function of max qps for our tested DNS servers on Unix systems:

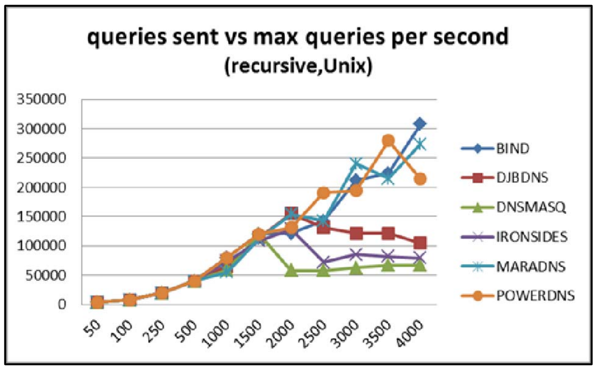

Fig. 6. Total queries sent vs max qps for recursive servers running Unix

Up to 1500 qps, the performance of the servers is essentially indistinguishable. At higher values, DNSMASQ, IRONSIDES and DJBDNS drop off fairly quickly. BIND was able to accept the most queries.

Similar results are seen in Fig. 7 for Windows systems, in that for smaller qps values the results are indistinguishable. Similar to the UNIX results, IRONSIDES drops off for higher qps, but interestingly so does WINDNS. BIND once again accepted the most queries, a total of about 302,000 over an 80-second run at 4000 qps.

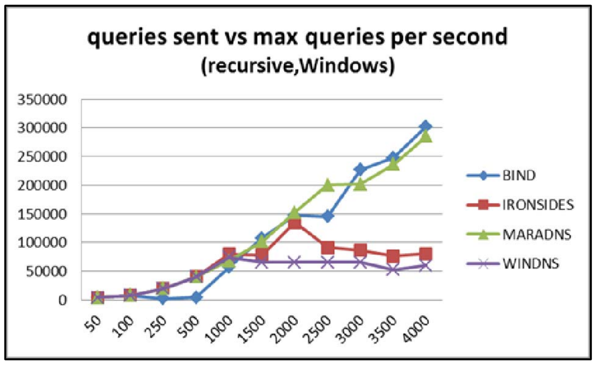

Fig. 7. Total queries sent vs max qps for recursive servers running Windows

2) Queries Lost: Figs. 8 and 9 below show the percentage of queries lost as a function of max qps for Unix and Windows systems. For this metric, lower is better. Values are rounded to one decimal place, so points that appear to be zero are actually small but non-zero values.

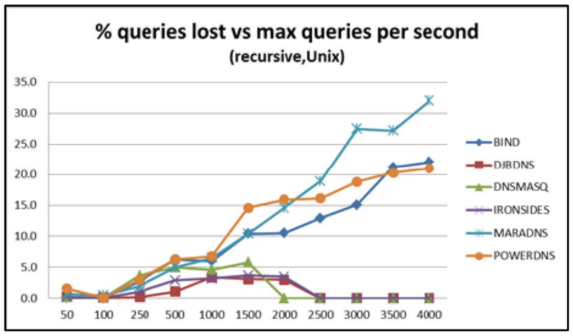

Fig. 8. % queries lost vs max qps for recursive servers running Unix

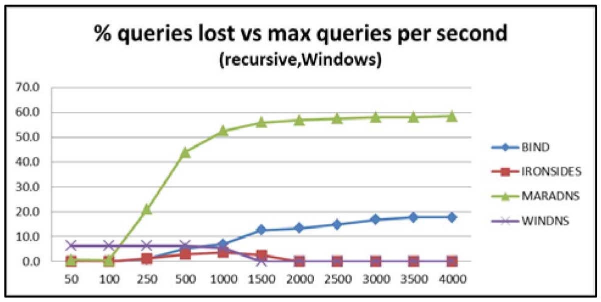

Fig. 9. % queries lost vs max qps for recursive servers running Windows

For Unix systems, the best performers in this area were DJBDNS, DNSMASQ, and IRONSIDES, all losing a miniscule percentage of queries at the highest loads tested. BIND did the worst at this metric, reflecting its policy of reporting SERVFAIL at its discretion to optimize its internal performance.

For Windows systems, the performance of WINDNS and IRONSIDES on this metric are indistinguishable, both dropping a very tiny percentage of queries at the highest loads tested. MARADNS dropped the most at about 60%, with BIND at 18%.

3) Successful Queries vs Max Queries Per Second: Figs. 10 and 11 show the number of successful queries versus max qps for Unix and Windows systems.

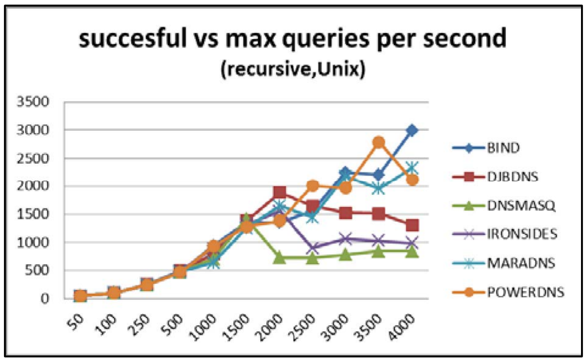

Fig. 10. Successful queries vs max qps for recursive servers running Unix

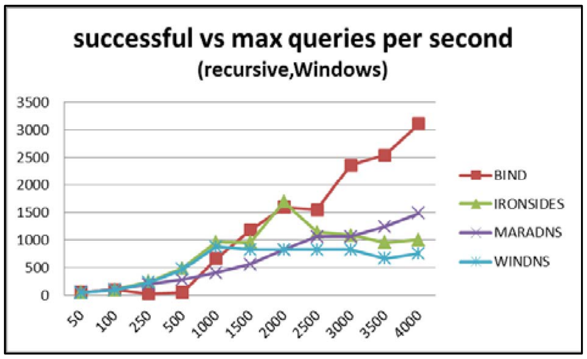

Fig. 11. Successful queries vs max qps for recursive servers running Windows

Once again, for this metric the performance of the servers at moderate loads is indistinguishable. On Unix systems, BIND does the best at higher loads, with IRONSIDES and DNSMASQ on the low end flattening out after 2500 qps. On Windows systems, IRONSIDES can keep pace with BIND for longer than WINDNS, dropping off at around 2500 qps. At high loads, IRONSIDES is significantly below BIND, but still above WINDNS.

4) Latency vs Max Queries Per Second: Figs. 12 and 13 show the average latency as a function of max qps for Unix and Windows systems, respectively. For this metric, smaller is better.

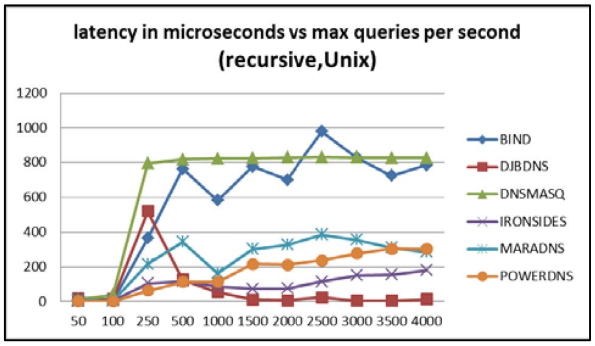

Fig. 12. Latency in microseconds for recursive servers running Unix

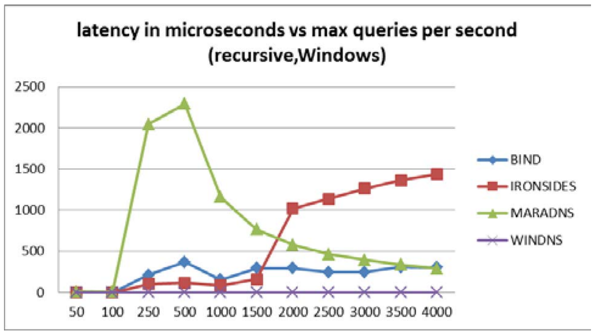

Fig. 13. Latency in microseconds for recursive servers running Windows

The latency for recursive servers was, as expected, significantly longer than that for authoritative servers, since initial access to an upstream server is required. For Unix systems, DJBDNS is incredibly fast: At 13 microseconds it's over an order of magnitude faster than its closest competitor IRONSIDES,

For Windows systems, as expected WINDNS performs the best. Clearly minimal DNS latency is important to Microsoft. The behavior of MARADNS is unusual in that latency peaks at relatively low qps values and then steadily improves to a stable value under higher loads. We were not able to determine the reason for this behavior. IRONSIDES has the highest latency, gradually increasing with workload. We presume this is due to its designed behavior of attempting to handle every query it can, at the cost of increased response time for individual queries.

V. CONCLUSIONS

As an authoritative server for Unix, IRONSIDES exhibited the best performance of all the servers we tested. For Windows, WINDNS performed better but only slightly, despite its status as a native Windows application developed by a multibillion dollar company.

As a recursive server, IRONSIDES can accept roughly as many queries as any other server we tested up to 1500-2000 qps, depending on the operating system. Beyond that, BIND performs the best, accepting approximately three times more queries at maximum loads tested. This is consistent with its design philosophy of responding to queries with SERVFAIL if it perceives the load is too high.

For Unix and Windows systems, the fraction of dropped queries is essentially indistinguishable between the top four servers, of which IRONSIDES is one. In terms of successfully processing queries under increasing loads, the performance of all servers including IRONSIDES was indistinguishable up to about 1500 qps. Under maximum loading, BIND performed the best and is significantly better than IRONSIDES, although IRONSIDES performed better than WINDNS under Windows.

In terms of latency, on Unix systems DJBDNS performed the best, over an order of magnitude faster than its closest competitor IRONSIDES. On Windows systems WINDNS is the fastest under all loading conditions. IRONSIDES is second up to about 1500 qps, but then degrades linearly up to the maximum tested value of 4000 qps.

Of all the servers tested, IRONSIDES is the only one with provable security properties. The closest competitor would be DJBDNS, with its offer of a $1000 payout for verified security problems. This bounty was eventually paid out in 2009 [18]. Clearly payout claims are no substitute for formal proofs of security.

We found no evidence that the use of formal methods in IRONSIDES required giving up performance. The performance of IRONSIDES was in most cases comparable and in some cases superior, depending on the loads tested. This is particularly significant given the relative resources expended in the development of IRONSIDES.

VI. Future Work

IRONSIDES does not yet offer the entire functionality of more widely used servers, and while its source is publicly available [19] it does not have support infrastructure. These are logical ways to add further value to IRONSIDES.

We hope this work will be further extended to apply formal methods and performance analysis outside the DNS domain, in the hopes of continued confirmation that internet software can be made provably more secure without significant sacrifices in performance. Web servers, for example, suffer from similar security problems for similar reasons. ICS and SCADA systems are currently attractive targets for hacking, and formal methods have been used to improve their security [20], but the effect of formal methods on performance in this domain remains unknown. These are the subject of current work at the Academy Center for Cyberspace Research.

Acknowledgment

This work was supported by the US Defense Advanced Research Projects agency and the Air Force Office of Scientific Research, under the auspices of the Academy Center for Cyberspace Research. The contributions of ACCR lab director Sean Harris are gratefully acknowledged. Finally, we thank the anonymous referees for their suggestions and feedback.

References

[1] https://tools.ietf.org/html/rfc882.

[2] Carnegie Mellon University Software Engineering Institute, “Multiple DNS implementations vulnerable to cache poisoning”, http://www.kb.cert.org/vuls/id/800113.

[3] Internet Security Consortium, BIND 9 Security Vulnerability Matrix, available online at https://kb.isc.org/article/AA-00913/0/BIND-9-Security-Vulnerability-Matrix.html

[4] https://technet.microsoft.com/library/security/MS15-127

[5] Barnes, J.: High Integrity Software: The SPARK Approach to Safety and Security. Addison-Wesley Publishing, 0-321-13616-0, © 2003.

[6] B. Fagin and M. Carlisle, “Provably secure DNS: A case study in reliable software,” 2013 International Conference on Reliable Software Technologies, Berlin, Germany pp 81-93.

[7] M. Carlisle and B. Fagin, “IRONSIDES: DNS with no single-packet denial of service or remote code execution vulnerabilities”, GLOBECOMM 2012, Anaheim CA.

[8] S. Gibson, Domain Name Server Benchmark, available online at https://www.grc.com/dns/benchmark.htm

[9] NameBench, https://namebench.en.softonic.com/.

[10] http://www.nominum.com/measurement-tools/

[11] Internet Services Simulation Suite, http://www.inetsim.org/.

[12] R. Mohan, “In defense of BIND: Open source DNS software yields a better breed of secure produt”, Security Week, May 25th 2010, available online at http://www.securityweek.com/defense-bind-open-source-dns-software-yields-better-breed-secure-product.

[13] https://www.isc.org/blogs/whats-your-version/

[14] https://cr.yp.to/djbdns/guarantee.html.

[15] Knot DNS: A high-performance authoritative-only DNS server, https://www.knot-dns.cz/

[16] S. Trenholme, “MaraDNS: A small open-source DNS server”, available online at http://maradns.samiam.org/

[17] https://www.powerdns.com/

[18] http://www.zdnet.com/article/dan-bernstein-confirms-djbdns-security-hole-pays-1000/

[19] http://ironsides.martincarlisle.com

[20] J. Groote, A. Osaiweran and J. Wsesselius, “Analyzing the effects of formal methods on the development of industrial control software”, 2011 IEEE Conference on Software Maintenance, Williamsburg VA, pp 467-472.

[21] H. Boulakhrif, “Analysis of DNS Resolver Performance Measurements”, Masters’ Thesis, Uuniversity of Amsterdam, 2015, available online at https://www.nlnetlabs.nl/downloads/publications/os3-2015-rp2-hamza-boulakhrif.pdf.