Formal Analysis of the DNS Bandwidth Amplification Attack and its Countermeasures Using Probabilistic Model Checking

Tushar Deshpande*, Panagiotis Katsaros†, Stylianos Basagiannis† and Scott A. Smolka*

*Department of Computer Science, Stony Brook University, Stony Brook, NY 11794-4400, USA
Email: {tushard, sas}@cs.stonybrook.edu

†Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece
Email: {katsaros, basags}@csd.auth.gr

***Abstract*—The DNS Bandwidth Amplification Attack (BAA) is a distributed denial of service attack in which a network of computers (zombies) flood a DNS server with responses to requests that have never been made. Amplification enters into the attack by virtue of the fact that a small 60-byte request can be answered by a substantially larger response of 4,000 bytes or more in size. We use the PRISM probabilistic model checker to introduce a Continuous Time Markov Chain model of the DNS BAA and three recently proposed countermeasures, and to perform an extensive cost-benefit analysis of the countermeasures. Our analysis, which is applicable to both DNS and DNSSec (a security extension of DNS), is based on objective metrics that weigh the benefits for a server in bandwidth usage against the cost incurred by incorrectly dropping legitimate traffic. The results we obtain, gleaned from more than 450 PRISM runs, demonstrate significant differences between the countermeasures as reflected by their respective net benefits. Our results also reveal that DNSSec is more vulnerable than DNS to a BAA attack, and, relatedly, DNSSec derives significantly less benefit from the countermeasures.**

***Keywords*-DDoS, Probabilistic Model Checking, DNS**

I. Introduction

DNS (Domain Name System) is a hierarchical naming system used to identify network hosts, so that a url (Uniform Resource Locator) can be used to address a machine in the internet. It is implemented using DNS name servers, which *resolve* resources (domains) into numeric IP addresses. DNS forms the logical backbone of the world wide web, and the service it provides is used on the order of a trillion times a day [12]. Any attack targeting DNS would thus seriously impact the security and basic operational status of the web and other services such as e-mail.

A Bandwidth Amplification Attack (BAA) is a distributed denial of service (DDoS) attack where a network of computers (zombies) is used to flood a DNS server—the *victim* of the attack—with excessively large DNS responses to requests that have never been made [27]. Moreover, a small 60-byte request can be answered by a substantially larger response of 4,000 bytes or more in size, serving to further imperil the victim.

A number of incidents involving BAA have been reported since 2002. An understanding of BAA *countermeasures* [9], [19], [32] and an assessment of their relative effectiveness are necessary for configuring today's DNS servers. Unfortunately, such an assessment has to date largely been lacking.

In this paper, we use the probabilistic model checker PRISM [14] to formally model and analyze the DNS BAA, along with three recently proposed countermeasures: *packet filtering*, *random packet drops*, and *aggressive retries* of legitimate packets. Moreover, we perform this analysis in the context of both DNS and DNSSec [6], a recently proposed security extension of DNS that uses digital signatures to authenticate the source of DNS data. DNSSec eliminates the possibility of DNS cache-poisoning attacks [12], [4], but has been criticized for potentially increasing the probability of DNS DDoS attacks [10].

Our approach is based on a Continuous Time Markov Chain (CTMC) model representing the bandwidth contention experienced by a victim server, while serving legitimate requests in the presence of the bogus traffic due to a BAA. The attack succeeds when a model state is reached where the available bandwidth (for processing legitimate DNS traffic) has been exhausted. As such, the *attack probability* can be computed by a probabilistic reachability property expressed in Continuous Stochastic Logic (CSL). For both DNS and DNSSec, model parameter values are established using data from recent DDoS incidents or measurements of amplification effects reported in the literature.

Our model possesses the fundamental Markov property, which is presumed in [11] and elsewhere: bandwidth sharing, under the common assumption of Poisson session arrivals, is insensitive to the flow size and the packet arrival process. Consequently, we can reasonably consider that the conditional probability distribution of future states depends only upon the present state.

Collectively, our results, gleaned from more than 450 runs of the PRISM model checker, formally validate the existence of the DNS BAA, and demonstrate a nonlinearly increasing attack probability with an increasing number of zombies. Our results also clearly show the effects of the proposed BAA countermeasures in terms of a reduction in the attack

probability. They also reveal that DNSSec is more vulnerable than DNS to a BAA attack, and, relatedly, DNSSec derives significantly less benefit from the countermeasures.

To better understand their relative strengths and weaknesses, we develop a number of CTMC reward-based properties in support of an extensive *cost-benefit analysis* of the BAA countermeasures. The analysis is conducted from the perspective of bandwidth usage by a victim server. Benefits include an increase in the percentage of legitimate packets processed by the server, and an increase in the period of time during which bandwidth is available. Cost measures include the percentage of legitimate traffic dropped as false positives, and percentage deviation from a target attack probability. We also consider the *computational cost* of a countermeasure.

The CTMC analysis we choose to perform is *transient* because the impact of a BAA in the steady-state is independent of the available bandwidth. Also, it is imperative to mitigate BAA effects within as short a period of time as possible [3]. Countermeasure parameters are set to the minimal values that result in a zero attack probability, while avoiding unnecessary countermeasure costs [7]. This allows for a fair comparison, demonstrating significant differences between the countermeasures, as reflected by their respective *net benefits*.

To the best of our knowledge, our CTMC model is the first probabilistic model for the DNS BAA and associated countermeasures. Moreover, our model-checking results and associated cost-benefit analysis may be viewed as a framework for comparing security threat levels and countermeasure effectiveness. Such a comparison may be useful in making an informed choice of countermeasure or in developing a defense policy that selects the optimal countermeasure for a BAA or related attack of varying strength.

The rest of this paper develops along the following lines. Section II outlines the principles of CTMC model checking. Section III describes the DNS BAA and countermeasures. Section IV introduces our PRISM CTMC model, and Section V presents our model-checking results and the results of our cost-benefit analysis. Section VI considers related work, while Section VII offers our concluding remarks.

II. Probabilistic model checking using CTMCs

CTMCs are widely used to model queuing systems, due to their strength in representing dynamic behavior and their advantage of being amenable to analytic treatment and numerical computation. Probabilistic model checking is an automated formal verification technique for modeling and analyzing systems or processes with probabilistic behavior, e.g. the CTMC of a queuing system. Model checking tools like PRISM [14] involve a combination of graph-theoretic algorithms for reachability analysis and iterative numerical solvers. Thus, it is possible to evaluate properties of the form $\mathcal{P}_{=?}(\phi)$ that compute the probability of some path that satisfies ϕ. Path formula ϕ is interpreted over the paths of the probabilistic model, which in our case is a CTMC. We define properties of the form `F prop`, where `F` is the "eventually" linear temporal operator and `prop` is a state assertion that evaluates to true or false for a single model state.

For our cost-benefit analysis, we take advantage of the CTMC rewards. CTMC allows us to attach rewards to *states* and *transitions* such that they get accumulated over time [21]. If at time t the CTMC has reached the n th state of some path, the reward is the sum of rewards accumulated in the preceding states (or transitions). Reward can be also interpreted as cost. In PRISM, the $\mathcal{R}$ operator is used for the analysis of reward-based properties. Four different types of reward-based properties are supported, but for our analysis we use only *cumulative reward* properties of the form ($\mathcal{C} \leq t$), since they are appropriate for evaluating the effects of BAA countermeasures, in the small time period during which the attack has to be mitigated.

A model in PRISM is constructed as the parallel composition of its modules. A module is described by a collection of commands, each of which comprises a guard and one or more update actions:

$$\texttt{[]}\ \texttt{g} \Rightarrow \lambda_1\ \texttt{:}\ \texttt{u}_1\ \texttt{+}\ \texttt{...}\ \texttt{+}\ \lambda_n\ \texttt{:}\ \texttt{u}_n\ \texttt{;}$$

The guard `g` is a predicate over model variables, whereas each $\texttt{u}_i$ describes by assigning new values to the variables, a transition that the module can make. For CTMCs, λ_i is the transition's rate, the parameter of a negative exponential distribution that governs the waiting time of the transition. If the guard is true, the updates are executed according to their rates. Commands can be labeled and this allows modules to interact with each other by synchronizing on identically labeled commands. In this case, the rate of the resulting transition is the product of the rates of the individual transitions.

III. The DNS BAA and Proposed Countermeasures

The DNS includes two types of servers, *name servers* and *resolvers*. DNS resolvers dispatch queries to name servers on behalf of clients that generate name-resolution requests. Once the resolver receives an answer it is forwarded to the client that originally generated the name-resolution request.

A. The DNS BAA

Let us assume that an attacker decides to attack the DNS resolver for the domain `cs.stonybrook.edu`, henceforth referred as the victim server.

1) The attacker prepares for the BAA by
   - Creating a query that would generate a substantially larger response called as amplification record.
   - Acquiring control of a large number of compromised hosts (zombies) to be used as attack sources.
   - Acquiring a list of resolvers.
2) The attacker drives the zombies to send requests for the amplification record to the previously found DNS resolvers. In these requests the zombies replace the source-address field with the victim's IP address.
3) The resolvers direct a very large numbers of excessively large DNS responses to the victim server, thereby exhausting victim's available bandwidth.

B. Why DNS and DNSSec are vulnerable to BAA?

DNS requests and responses are sent as UDP packets, and are thus vulnerable to source-address spoofing. Also, a request can generate a substantially larger response, such that the ratio of the response size divided by the size of the request, termed as *amplification factor* (AF) can be as high as 73 [19], [31].

DNSSec prevents cache-poisoning attacks, but it does not provide protection against IP spoofing [26]. Some additional records are used in order to authenticate responses and this increases bandwidth consumption. For each DNSSec request a chain of responses [10] may be returned. For a full implementation of DNSSec and a chain of eight responses for each request, it is possible to have an AF of 271.2.

C. Countermeasures against DNS BAA

In the DNS, typical BAAs have been found to scale up to 2.8 Gbps. Some massive-scale attacks have also been reported that exploited around 140000 name servers resulting in a volume of 10 Gbps [19]. Any attempt to add more resources (server, bandwidth etc.), in order to handle the increased BAA load [15] has therefore a limited scope for improvement and this necessitates the study of countermeasures that mitigate the attack actively. We study three different countermeasures that either aim to reduce the attack traffic or alternatively to generate redundant legitimate traffic [32].

- **Filtering (FTR)** Filtering tries to identify the attack sources and block the traffic originating from them. A relatively high packet arrival rate or detection of unusually large sized packets can indicate presence of a BAA [15]. From the algorithms described in [29], [24], [18], [37], [16], [2], [35] we conclude that the filtering offers relatively high accuracy with false positives as low as 10%. Computational cost of FTR depends on the filtering mechanism and the attack strength.
- **Random Drops (RND):** This approach regulates the incoming traffic by randomly dropping UDP packets [22], [17]. During a BAA, the traffic arriving at the victim mainly comprises of the bogus packets and a randomly dropped packet is likely to be a bogus packet, thus mitigating the attack with negligible computational costs.
- **Aggressive Retries (AGR):** Trusted clients are encouraged to generate traffic at a doubled rate during each retry [32], [20], so that the proportion of legitimate traffic is increased. Downside is the additional server workload and the increased bandwidth consumption.

IV. The PRISM Model for the BAA

Our PRISM model for the BAA comes in four versions, one without any countermeasure in place (henceforth referred as the no-fix case), and one for each countermeasure.

A. The PRISM Model for the DNS BAA

Our model defines two primary modules.

- **Client Server (CS)**: CS represents the victim server. A finite capacity queue emulates the bandwidth of the CS. This bandwidth is shared by legitimate DNS queries and response, and the BAA traffic. When the queue becomes full, the available bandwidth is exhausted and all subsequently arriving packets are dropped.
- **Net**: Net provides an abstracted representation of all DNS resolvers and clients that generate legitimate and attack traffic for the CS.

Two additional modules are used for modeling FTR and RND. Module **Filter** emulates a packet-filtering mechanism and module **RandomDropper** provides a random packet-dropping mechanism.

The model parameters are as follows:

- `bogus_rate`: the rate at which each zombie generates spoofed bogus UDP requests. Since, the attack strength mainly depends on the number of zombies, we use a moderate `bogus_rate` of 10 requests per sec.
- `zombies`: the number of zombies used in the attack.
- `UDP_rate`: the rate at which legitimate UDP request and reply packets arrive at the CS. Since bandwidth sharing depends on the attack strength, we consider a fixed arrival rate of 100 packets per second.
- `BandwidthQueueUpperLimit (BQUL)`: the capacity of the bandwidth queue in terms of number of packets. The maximum dedicated bandwidth required for a typical DNS server can be approximated to 1 Mbps [36]. The average packet size for DNS is 286 bytes (with request size of 60 bytes and response size of 512 bytes [31]). The `BQUL` for DNS is $\frac{1Mb}{286} = 458$ packets.
- `effective_amplification_factor (EAF)`: the ratio of bogus response packet size to the average packet size of the incoming legitimate traffic. For any DNS server the legitimate traffic contains not only requests but also responses. To account for the responses included in the legitimate traffic, we use EAF instead of AF. E.g. the maximum packet size of a bogus DNS response can be $60 * 73 = 4380$ bytes, where 73 is the AF for DNS [31]. The EAF for DNS is then computed as $\frac{4380}{286} = 15.31$.
- `serve_rate`: the rate at which the CS serves the queued UDP packets. Using data from case study [30], we set `serve_rate` to 12666, but as discussed later this parameter does not affect the ranking of the three countermeasures in terms of their performance.

The countermeasure parameters are:

- `detection_percentage (dp)`: the percentage of attack traffic identified and filtered by FTR
- `false_positive_percentage (fpp)`: the percentage of legitimate traffic incorrectly categorized as bogus by FTR
- `random_drop_percentage (rdp)`: the percentage of incoming UDP packets randomly dropped by RND
- `retries`: the number of times that requests are resent, to increase the share of legitimate traffic by AGR

Each module defines certain actions, which synchronize with actions in the other modules that share the same label. Each action has an associated rate. Actions also have associ-

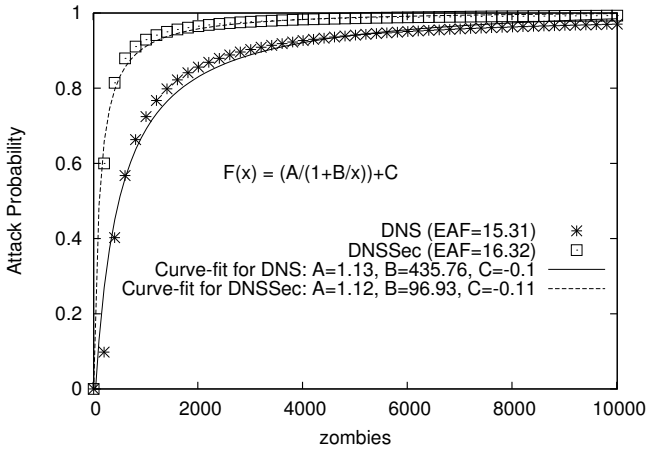

Fig. 1. Effect of BAA on DNS/DNSSec Nofix

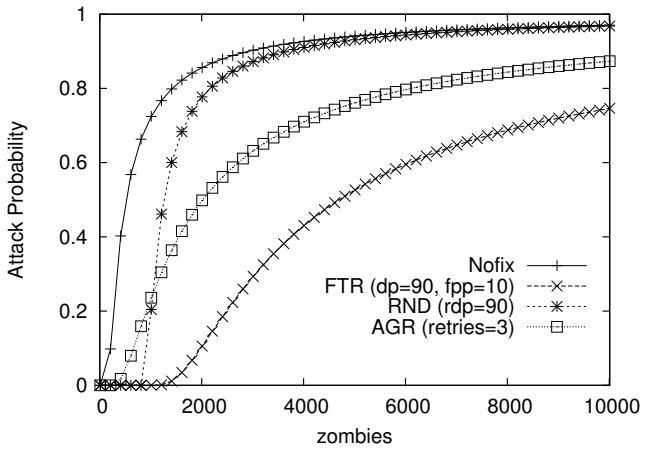

Fig. 2. Countermeasure Effect on Attack Probability for DNS

ated preconditions that need to be satisfied for their execution to take place. For synchronized actions, the effective rate is the product of rates of the individual actions. We now describe some of the important actions for each module.

- **Receive UDP packet**: A legitimate UDP packet is received by the CS and a position in CS's bandwidth queue is occupied. This action is defined in modules **CS** and **Net** and occurs at the rate: (i) `UDP_rate` for the no-fix case, (ii) $\texttt{UDP\_rate} \cdot (1 - \texttt{fpp}/100)$ for FTR, (iii) $\texttt{UDP\_rate} \cdot (1 - \texttt{rdp}/100)$ for RND, (iii) $(\texttt{UDP\_rate}/2) \cdot (1 + 2^{\texttt{retries}})$ for AGR.
- **UDP packet lost**: A UDP packet is dropped, because CS's bandwidth is full. This action is defined in modules **CS** and **Net** and it occurs at the same rates with the action *Receive UDP packet*.
- **Receive bogus packet**: Action defined in modules **CS** and **Net** representing receipt of bogus packets by the CS and corresponding reduction of the available bandwidth. For the no-fix case this action occurs at an overall rate of $\texttt{bogus\_rate} \cdot \texttt{zombies} \cdot \texttt{EAF}$. For the countermeasure models, rates are adjusted (except AGR) as in **Receive UDP packet** by using parameters `dp` and `rdp`.
- **Bogus packet lost**: Action defined in modules **CS** and **Net**. It is the analogue of action **UDP packet lost** for the bogus UDP packets.
- **Serve UDP Packet**: Action defined in CS representing the service of packets in the bandwidth queue at a rate `serve_rate`.
- **Client Request**: Action defined in modules **CS** and **Net** for dispatching a client request to the CS with rate `UDP_rate`. This action can be executed if the bandwidth is available. After having been executed once, this action is permanently disabled. It is used as a handle for computing the attack probability.

B. The PRISM Model for the DNSSec BAA

The PRISM model for the BAA over DNSSec differs only in the parameter values assigned for `BQUL` and `EAF`.

A typical 70 bytes request in DNSSec can generate a legal response of 2257 bytes [10] and a reasonable value for the average size of legal packets is therefore $\frac{70+2257}{2} = 1163.5$ bytes. For a server having bandwidth equal to the server considered in the DNS case, we derive the value for the parameter `BQUL` in DNSSec as $\frac{1Mb}{1163.5} = 112$ packets. If we assume a BAA with bogus DNSSec response chains of length 8 and size 18984 bytes [10], i.e. a not particularly strong BAA, we get an $\texttt{EAF} = \frac{18984}{1163.5} = 16.32$, comparable to the one used for the DNS.

V. Experimental Results

A. Impact of BAA on DNS and DNSSec

We compute attack probability using CSL formula `P=? [F DenialOfService]` with predicate `DenialOfService` becoming true, when the **Client Request** action is not completed by the CS under attack. Fig. 1 shows the impact of BAA on DNS and DNSSec in the absence of any countermeasure. The attack probability for DNSSec is consistently higher than that for DNS and this result formally validates the claim of [10] that DNSSec is more vulnerable to the BAA. The same figure also shows the results of curve fitting to the computed attack probabilities. We observe that, with an increasing number of zombies x, the attack probability $F(x)$ increases nonlinearly according to the function $\frac{A}{1+(B/x)} + C$.

Fig. 2 shows the countermeasure effects on the BAA probability for DNS. All countermeasures, if configured with the shown parameters reduce the BAA probability, with a more important improvement for FTR, followed by AGR and RND.

B. Cost-benefit analysis for BAA countermeasures

1) Need for cost-benefit analysis: Fig. 2 indicates significant reductions in the BAA probability for the particular countermeasure parameters. However, these results alone cannot be the basis for a formal comparison between the countermeasures, since they follow different approaches to mitigate BAAs and depend on different parameters. Comparison of countermeasures cannot be based on the BAA probability for a client, because by selecting proper countermeasure settings, this probability can be made arbitrarily small or even zero. All benefits and costs associated with a countermeasure are experienced by the victim server and this is the reason for

zombies	RND		AGR	
	rdp		retries	
	DNS	DNSSec	DNS	DNSSec
100		24		3
500	82	86	4	7
900	90.5	92.1	5	8
4000	98	98.26	8	10
8000	99	99.17	9	11

TABLE I
COUNTERMEASURE PARAMETERS FOR RND AND AGR TO ACHIEVE *zero attack probability*

evaluating the countermeasures from the server's perspective. To set a ground for comparison, we run our model to achieve zero BAA probability while using minimal values of countermeasure parameters to avoid unnecessary countermeasure costs. This is a typical application of our proposal in [7] for repairing probabilistic models. Our analysis is limited to costs and benefits that refer to bandwidth usage and does not take into account other important aspects like computational or economic cost that we discuss separately at the end. To ensure the validity of the results, we chose the cost-benefit metrics such that they do not compute the same quantity twice while at the same time they address the full set of effects of a countermeasure on bandwidth usage. For the cost-benefit analysis to be meaningful, the metrics should allow for a uniform interpretation of costs and benefits, thus making it possible to aggregate the results into a single net-benefit value.

2) Defining benefit and cost metrics: All metrics are evaluated as percentage values (no units), so as to compute the net benefit of a countermeasure [33].

- **Percentage B_1 of legitimate packets in total packets processed:** An ideal countermeasure attempts to increase B_1 and it is therefore considered a benefit metric.
- **Percentage time B_2 spent in states where bandwidth is available:** A server is benefited if bandwidth availability is conserved for longer time periods.
- **Percentage C_1 of legitimate packets dropped:** False positive-based costs are observed for FTR and RND, but not for AGR.
- **Percentage deviation C_2 from desired attack probability:** Every FTR algorithm [29], [24], [18], [37], [16], [2], [35] is characterized by specific values for the `dp` and `fpp` and it is not possible to achieve a zero attack probability for high-volume BAAs. Such a countermeasure can still be a cost-efficient solution if the attack probability is within an acceptable range. We capture this situation in our analysis by introducing another cost which is defined as the deviation of the observed attack probability from the desired value.

Net Benefit: For a countermeasure, $net\ benefit = (B_1 + B_2 - C_1 - C_2)$ [34]. The countermeasure offering the highest net benefit is the most cost-effective.

3) Computing benefits and costs: A BAA that is sustained for a relatively longer duration is easy to detect and thus, attackers prefer to launch attacks in bursts [3]. It is therefore imperative to mitigate the effects of a BAA within a short period of time. This is the main reason for using cumulative reward properties that quantify accumulated rewards in the short-term up to a given time-bound, as opposed to the steady-state alternative [25]. From our experiments we also realized that, the impact of the BAA in the steady-state becomes independent of the available bandwidth, and the relative differences in the countermeasure effects are flattened. We define three reward properties P_1, P_2 and P_3 of the form `R{"<reward definition>"}=? [C<=t]`, that evaluate the accumulated quantities for rewards R_1, R_2, R_3 as follows.

R_1 is a transition reward that assigns a unit reward to actions *Receive UDP packet* and *Client Request*. So, property P_1 counts the total number of legitimate packets received, say PK_1. R_2 is another transition reward with unit reward attached to the action *Receive bogus packet*. The property P_2 counts the total number of bogus packets received, say PK_2. Then, $B_1 = \frac{PK_1}{PK_1+PK_2} \cdot 100$.

Reward R_3 is a state reward that assigns a unit reward per unit of time spent in the states where the victim's bandwidth is not full. Therefore, property P_3 yields the total time T_1 spent in states where bandwidth is available. Then, $B_2 = \frac{T_1}{t} \cdot 100$, where t is the time duration for which the countermeasure effects are evaluated which is same as the time-bound used for evaluating P_3.

Due to false positives, FTR drops `fpp`% of the incoming legitimate traffic. So, the cost C_1 for FTR is `fpp`. The percentage of incoming traffic dropped by RND is given by `rdp`. Since incoming traffic includes bogus as well as legitimate traffic, RND drops `rdp`% of incoming legitimate traffic, thereby producing false positives. The cost C_1 for RND thus equals `rdp`. For AGR, C_1 is *zero*.

The cost C_2 is *zero* for RND and AGR, for which parameters `rdp` and `retries` can be in theory adjusted, such that any desired attack probability can be achieved. For FTR, we compute C_2 by recording the percentage deviation from the desired attack probability, if any.

4) Determining countermeasure parameters: Studies of different filtering algorithms show that on average FTR has a `dp` of 90% and `fpp` of 10%. While these values depend on the accuracy of the algorithm, they remain unchanged for different rates of incoming traffic. Since FTR has fixed `dp` and `fpp`, we cannot always achieve *zero attack probability* for it. However, the parameters of AGR and RND can always be adjusted such that the attack probability becomes zero.

Table I shows the countermeasure parameters required to achieve *zero attack probability* for RND and AGR with the number of zombies varied from 10 to 10000 so that we obtain 28 distinct values. For brevity, only 5 representative values for `zombies` are shown in Table I. For both DNS and DNSSec, a countermeasure was enabled only when a non-zero attack probability was observed for the no-fix case. In the DNS case, this happened for `zombies` ≥ 200 whereas for DNSSec, the countermeasures were enabled when `zombies` ≥ 100. The empty cells indicate that a countermeasure was not enabled

zombies	FTR								RND						AGR			
	DNS				DNSSec				DNS			DNSSec			DNS		DNSSec	
	B1	B2	C1	C2	B1	B2	C1	C2	B1	B2	C1	B1	B2	C1	B1	B2	B1	B2
100					5.8	100	10	0				0.7	100	24			2.8	81.8
500	1.3	100	10	0	1.2	100	10	0	0.2	100	82	0.2	100	86	1.2	22.3	7.4	15.6
900	0.7	100	10	0	0.7	92.4	10	2.4	0.1	100	90.5	0.1	100	92.1	1.2	12.4	8.1	8.6
4000	0.2	28.1	10	42.9	0.2	21.1	10	82.6	0	100	98	0	100	98.3	2.1	2.8	7.3	2.0
8000	0.1	14.1	10	68.7	0.1	10.5	10	91.8	0	100	99	0	100	99.2	2.1	1.4	7.3	1

TABLE II
COST AND BENEFIT VALUES FOR FTR, RND AND AGR

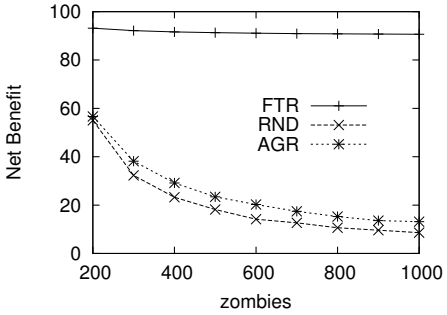

Fig. 3. Net benefit for DNS with zombies varied from 200 to 1000

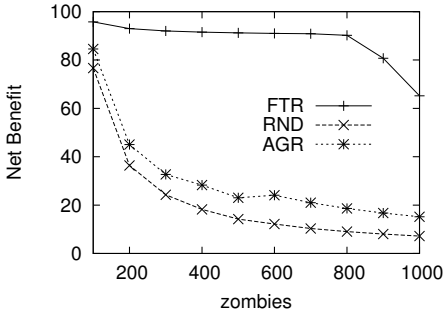

Fig. 5. Net benefit for DNSSec with zombies varied from 100 to 1000

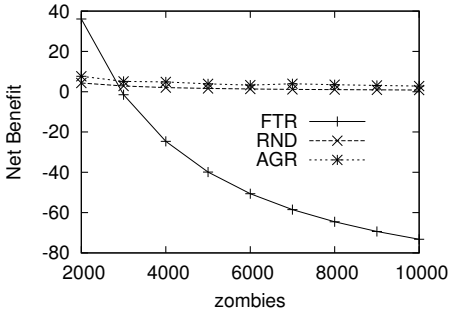

Fig. 4. Net benefit for DNS with zombies varied from 2000 to 10000

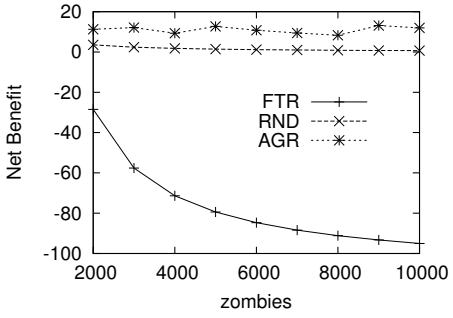

Fig. 6. Net benefit for DNSSec with zombies varied from 2000 to 10000

for the corresponding number of zombies. We report `rdp` for RND and `retries` for AGR.

From Table I, we see that as the attack strength increases, the required countermeasure strength increases too. We also observe that higher countermeasure parameter values are required for DNSSec when compared with the parameter values for DNS. This validates our result from Fig. 1 that DNSSec is more vulnerable than DNS to a BAA.

5) Performing cost-benefit analysis: Using the countermeasure parameter values from Table I and the PRISM reward properties that were previously discussed, we compute B_1, B_2, C_1 and C_2 for FTR, RND and AGR.

The time-bound t for the cumulative reward properties is selected such that it allows the model dynamics to reasonably evolve, while at the same time prevents it from attaining steady-state (where effects on bandwidth usage are independent of the available bandwidth). An appropriate time-bound $t = 0.1$ was found by experimentation.

Table II reports the costs and benefits for FTR, RND and AGR. Empty cells indicate that a countermeasure was not enabled for the corresponding number of zombies.

Figs. 3-6 show the net benefit for FTR, RND and AGR computed using results from Table II. We observe that,

- For DNS and DNSSec, the net benefits of FTR, RND and AGR decrease as more zombies participate in the attack.
- From Figs. 3-4, we observe that for DNS, FTR is more cost-effective when `zombies` ≤ 2000. Thereafter, AGR offers the highest net benefit. The net benefit of FTR rapidly drops for `zombies` > 2000, since, as seen from Table II, FTR's C_2 increase to cancel out the benefits. A similar behavior is observed for DNSSec from Figs. 5-6, where FTR is the best countermeasure when `zombies` ≤ 1000. Thereafter, AGR offers the highest net benefit.
- From Figs. 3, 5 and Figs. 4, 6, we observe that FTR and RND offer higher net benefit for DNS than the net benefit they offer for DNSSec. Similarly, for `zombies` ≤ 500, AGR's net benefit is higher for DNS than DNSSec. This once again highlights that compared to DNS, DNSSec is more vulnerability to a BAA.
- When `zombies` > 500, AGR offers more net benefit to

DNSSec than the net benefit it offers to DNS, suggesting that compared to FTR and RND, AGR scales well for increasing attack strengths. AGR works by prompting the clients to resend the legitimate packets at a doubled rate for each retry. The average size of a legal DNSSec packet is 1163.5 bytes as compared to an average DNS legal packet with size 286 bytes. So, AGR increases volume of legitimate traffic for DNSSec far more rapidly as compared to DNS.

C. Computational Cost of a Countermeasure

The computational cost of a countermeasure is reflected in the amount of computing resources (e.g. processors) required by the countermeasure. While the computational cost increases infrastructure cost, it does not affect the DNS server's performance, since typically a countermeasure is supported on a separate piece of hardware; e.g. FTR is implemented by firewalls. However, computational cost must be considered while deploying a countermeasure since this cost can offset any gains obtained in bandwidth usage. We now estimate computational cost for FTR, which can be potentially high compared to the cost for the two other countermeasures.

The computational cost for FTR is calculated based on the aggregate incoming traffic that has to be filtered and can thus be determined without using our PRISM model. By applying curve-fitting to the data reported in [16] we find that for the xPF filtering algorithm, the number of instructions executed per packet is $\frac{(4.95 \cdot 10^8)}{\texttt{par}} + 6123.14$, where `par (packet arrival rate)` is the rate at which IP packets arrive at the server. For a generic 4-stage instruction pipeline, 4 instructions execute in 7 cycles. Thus FTR needs $\frac{(8.7 \cdot 10^8)}{\texttt{par}} + 10715$ processor cycles per packet. For a 733 MHz processor [30], this translates to $(1.2 + 1.4 \cdot 10^{-5} \cdot \texttt{par})$ processors required to sustain FTR performance for packets arriving at rate `par`. The processor utilization by FTR is thus directly proportional to the `par` and in turn to the number of zombies, leading to potentially very high computational cost if a large number of zombies participate in the attack.

D. Significance and limitations of cost-benefit analysis

Although, we performed experiments with a single value of `serve_rate`, the conclusions of our cost-benefit analysis remain valid even if the `serve_rate` changes. As an example, our results show that the AGR offers higher net benefit than RND. If we increase `serve_rate`, then for both RND and AGR B_1 tends to zero, B_2 tends to 100 and for RND C_1 tends to zero. This causes the net benefit of AGR to remain always higher than the net benefit of RND.

In general, we expect computational costs for RND and AGR to be lower than the computational cost for FTR. However, to increase our confidence for the countermeasure cost-effectiveness in BAAs of very high volume, we need to precisely measure the computational costs since a high computational cost may hinder one from deploying a countermeasure solution with zero attack probability for high volume BAAs.

Our cost-benefit analysis can be potentially applied to other network security problems, which can be efficiently analyzed by probabilistic model checking. Any similar approach should use valid and meaningful cost and benefit metrics, according to the principles mentioned in section V-B.

VI. Related Work

We presented a CTMC-based analysis of the DNS BAA and its countermeasures by probabilistic model checking. Our analysis formally validates the existence of the DNS BAA and defines CTMC reward properties that allow a comparison between the costs and the benefits of the proposed countermeasures, in terms of bandwidth usage by a DNS server.

In related work, there are some proposals for the formal analysis of DoS attacks by model checking, which differ from our analysis in several ways. However, there is no formal analysis for the DNS BAA. In [1], the authors use probabilistic rewriting logic to model a DoS-resistant 3-way handshaking in the Transmission Control Protocol. Instead of using a formal stochastic model like a CTMC, they generate a timed probabilistic model from the developed algebraic specification, which is then analyzed by statistical model checking. This is a simulation-based analysis and consequently it cannot be as accurate as the probabilistic model checking approach. Statistical model checking is also used in [5], in order to analyze the ASV protocol as a DoS countermeasure. Cost and benefit metrics based on rewards attached to the underlying formal model are not supported.

Another related work is [8], which presents a Discrete Time Markov Chain model for a DoS threat against an authentication protocol. The cost-based analysis framework of [23] is instantiated in a probabilistic model checking framework that compares the cost incurred to the attacker against the cost incurred to the honest participants during a DoS. CTMC based analysis with reward properties is proposed in [13], as a mean to analyze a DDoS attack against the Mobile IP and Seamless IP diversity based Generalized Mobility Architecture. In both [8], [13] no analysis and comparison of related countermeasures is provided.

Finally, in [28] a characterization of an amplification attack is introduced by means of states where some measure comparisons hold true. This is called measure checking and is implemented with rewriting logic, an executable specification that is model checked in the Maude tool. This approach focuses on automatically finding known attacks and verifying that a patch for an attack achieves its aim, but no comparison between alternative solutions is supported.

VII. Conclusions

We used the PRISM probabilistic model checker to formally model and analyze the DNS BAA and its countermeasures. The nature of the proposed countermeasures (packet filtering, random packet drop, aggressive retries) made BAA an ideal candidate for probabilistic model checking. Moreover, since

the BAA is aimed at DNS servers, it was both natural and beneficial to model it in PRISM as a CTMC, with corresponding arrival rates for benign and malicious DNS traffic.

We provided model-checking results for the attack probability which clearly demonstrate that DNSSec is more vulnerable than DNS to a BAA. Our curve-fitting results show a nonlinear increase in attack probability with an increasing number of zombies. We also presented a flexible cost-benefit analysis of BAA countermeasures that allows one to find the most cost-effective countermeasure for attacks of different strengths.

As future work, we intend to extend our BAA countermeasure cost-benefit analysis to the BAA countermeasures that combine FTR, RND and AGR. The challenge in this setting would be to determine the optimum values for multiple countermeasure parameters (`dp`, `fpp`, `rdp` and `retries`) to achieve the desired attack probability while maximizing benefit and minimizing cost.

The source files for our PRISM model of the DNS Bandwidth Amplification Attack are available from http://www.cs.stonybrook.edu/~sas/baa/.

REFERENCES

[1] G. Agha, M. Greenwald, C. A. Gunter, S. Khanna, J. Meseguer, K. Sen, and P. Thati. Formal modeling and analysis of DoS using probabilistic rewrite theories. In *Proc. IEEE Workshop on Foundations of Computer Security*, 2005.

[2] U. Akyazi and A. Uyar. Detection of DDoS attacks via an artificial immune system-inspired multiobjective evolutionary algorithm. In *Applications of Evolutionary Computation*, volume 6025 of *LNCS*, pages 1–10. 2010.

[3] B. Al-Duwairi. *Mitigation and traceback countermeasures for DDoS attacks*. PhD thesis, 2005.

[4] N. Alexiou, T. Deshpande, S. Basagiannis, P. Katsaros, and S. A. Smolka. Formal analysis of the Kaminsky DNS cache-poisoning attack using probabilistic model checking. In *IEEE International Symposium on High-Assurance Systems Engineering*, pages 94–103, 2010.

[5] M. AlTurki, J. Meseguer, and C. A. Gunter. Probabilistic modeling and analysis of DoS protection for the ASV protocol. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 234:3–18, Mar 2009.

[6] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. RFC 4035: Protocol modifications for the DNS security extensions. RFC 4035 (Informational), March 2005.

[7] E. Bartocci, R. Grosu, P. Katsaros, C. R. Ramakrishnan, and S. A. Smolka. Model repair for probabilistic systems. In *Proc. 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'11)*, volume 6605 of *LNCS*, pages 326–340, 2011.

[8] S. Basagiannis, P. Katsaros, A. Pombortsis, and N. Alexiou. Probabilistic model checking for the quantification of DoS security threats. *Computers and Security*, 28(6):450–465, Sep 2009.

[9] S. Changhua, L. Bin, and S. Lei. Efficient and low-cost hardware defense against DNS amplification attacks. In *Proc. IEEE Global Telecommunications Conference, 2008 (IEEE GLOBECOM '08)*, pages 1–5, Dec 2008.

[10] A. Cowperthwaite and A. Somayaji. The futility of DNSSec. In *Proc. 5th Annual Symposium on Information Assurance (ASIA '10)*, pages 2–8, 2010.

[11] S. Ben Fred, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. *ACM SIGCOMM Computer Communication Review*, 31(4):111–122, 2001.

[12] S. Friedl. An illustrated guide to the Kaminsky DNS vulnerability. *Unixwiz.net Tech Tips*, Aug 2008.

[13] S. Fu and M. Atiquzzaman. Survivability evaluation of SIGMA and mobile IP. *Wireless Personal Communications: An International Journal*, 43:933–944, Nov 2007.

[14] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '06)*, volume 3920 of *LNCS*, pages 441–444, 2006.

[15] ICANN. Factsheet: Root server attack on 6 february 2007. Technical report, Internet Corporation for Assigned Names and Numbers (ICANN), Mar 2007.

[16] S. Ioannidis, K. G. Anagnostakis, J. Ioannidis, and A. D. Keromytis. xPF: packet filtering for low-cost network monitoring. In *Workshop on High Performance Switching and Routing Merging Optical and IP Technologies, 2002*, pages 116–120, 2002.

[17] V. Jacobson. Minutes of the performance working group. In *Proc. the Cocoa Beach Internet Engineering Task Force*, Apr 1989.

[18] C. Jin, H. Wang, and K. G. Shin. Hop-count filtering: an effective defense against spoofed DDoS traffic. In *Proc. 10th ACM conference on Computer and communications security (CCS'03)*, pages 30–41, 2003.

[19] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis. Detecting DNS amplification attacks. In *Critical Information Infrastructures Security*, volume 5141 of *LNCS*, pages 185–196. 2008.

[20] S. Khanna, S. S. Venkatesh, O. Fatemieh, F. Khan, and C. A. Gunter. Adaptive selective verification. In *Proc. 27th IEEE Conference on Computer Communications (INFOCOM '08)*, pages 529–537, Apr 2008.

[21] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *LNCS (Tutorial Volume)*, pages 220–270, 2007.

[22] A. Mankin. Random drop congestion control. In *Proc. of the ACM symposium on Communications architectures & protocols*, SIGCOMM '90, pages 1–7, 1990.

[23] C. Meadows. A formal framework and evaluation method for network denial of service. In *Proc. of the 12th IEEE workshop on Computer Security Foundations (CSFW'99)*, 1999.

[24] T. Peng, C. Leckie, and K. Ramamohanarao. Protection from distributed denial of service attacks using history-based IP filtering. In *Proc. IEEE International Conference on Communications, 2003 (ICC'03)*, volume 1, pages 482–486, May 2003.

[25] PRISM manual (http://www.prismmodelchecker.org/manual/Main/Welcome).

[26] A. Sagar. DNSSec: A vision. Presented in ICANN's 31st International Public Meeting, New Delhi, India, Feb 2008.

[27] Security and Stability Advisory Committee. DNS distributed denial of service (DDoS) attacks. *SSAC Advisory SAC008*, 2006 (http://www.icann.org/en/committees/security/dns-ddos-advisory-31mar06.pdf).

[28] R. Shankesi, M. AlTurki, R. Sasse, C. Gunter, and J. Meseguer. Model-checking DoS amplification for VoIP session initiation. In *European Symposium on Research in Computer Security*, pages 390–405, 2009.

[29] H. Sun, Y. Zhaung, and H. J. Chao. A principal components analysis-based robust DDoS defense system. In *Proc. IEEE International Conference on Communications, 2008 (ICC'08)*, pages 1663–1669, May 2003.

[30] Monitoring DNS server performance (http://technet.microsoft.com/en-us/library/cc778608(WS.10).aspx). Technical report.

[31] R. Vaughn and G. Evron. DNS amplification attacks. Technical report, 2006 (http://www.isotf.org/news/DNS-Amplification-Attacks.pdf).

[32] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker. DDoS defense by offense. *ACM Transactions on Computer Systems*, 28:3:1–3:54, Aug 2010.

[33] T. Watkins. An introduction to cost benefit analysis (http://www.sjsu.edu/faculty/watkins/cba.htm).

[34] Cost-benefit analysis (http://en.wikipedia.org/wiki/Cost-benefit_analysis).

[35] Z. Wu, M. Xie, and H. Wang. Swift: a fast dynamic packet filter. In *Proc. 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*, pages 279–292, 2008.

[36] Y. Xuebiao, W. Xin, L. Xiaodong, and Y. Baoping. DNS measurements at the .CN TLD servers. In *Proc. Sixth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD '09)*, volume 7, pages 540–545, Aug 2009.

[37] W. Yuhui. Hybrid traceback-filtering (HTF): An efficient DoS/DDoS defense mechanism. Master's thesis, National University of Singapore, 2005.