Resource-Bounded Intruders in Denial of Service Attacks

Abraão Aires Urquiza[*], Musab A. AlTurki[†‡], Max Kanovich[§¶],
Tajana Ban Kirigin[||], Vivek Nigam[x*], Andre Scedrov[††¶] and Carolyn Talcott[‡‡]

[*] Federal University of Paraíba, Brazil, Email: abraauc@gmail.com
[†] KFUPM, Dhahran, Saudi Arabia,Email: musab@kfupm.edu.sa
[‡]Runtime Verification Inc., USA
[§] University College London, UK, Email: m.kanovich@ucl.ac.uk
[¶] National Research University Higher School of Economics, Moscow, Russia
[||] University of Rijeka, Department of Mathematics, HR, Email: bank@math.uniri.hr
[x] fortiss, Germany, Email: nigam@fortiss.org
[††]University of Pennsylvania, USA, Email: scedrov@math.upenn.edu
[‡‡]SRI International, USA, Email: clt@csl.sri.com

***Abstract*—Denial of Service (DoS) attacks have been a serious security concern, as no service is, in principle, protected against them. Although a Dolev-Yao intruder with unlimited resources can trivially render any service unavailable, DoS attacks do not necessarily have to be carried out by such (extremely) powerful intruders. It is useful in practice and more challenging for formal protocol verification to determine whether a service is vulnerable even to resource-bounded intruders that cannot generate or intercept arbitrary large volumes of traffic. This paper proposes a novel, more refined intruder model where the intruder can only consume at most some specified amount of resources in any given time window. Additionally, we propose protocol theories that may contain timeouts and specify service resource usage during protocol execution. In contrast to the existing resource-conscious protocol verification models, our model allows finer and more subtle analysis of DoS problems. We illustrate the power of our approach by representing a number of classes of DoS attacks, such as, Slow, Asymmetric and Amplification DoS attacks, exhausting different types of resources of the target, such as, number of workers, processing power, memory, and network bandwidth. We show that the proposed DoS problem is undecidable in general and is PSPACE-complete for the class of resource-bounded, balanced systems. Finally, we implemented our formal verification model in the rewriting logic tool Maude and analyzed a number of DoS attacks in Maude using Rewriting Modulo SMT in an automated fashion.**

I. Introduction

For the past decades, Denial of Service (DoS) attacks and their distributed version (DDoS) have been a serious security concern, as no service is, in principle, protected against them. Indeed, if the intruder, such as the Dolev-Yao (DY) intruder [15], has enough resources, he may render any service unavailable by sending a large number of messages (flooding attacks) or by intercepting all messages in the network (jamming attacks). What makes the DoS threat even worse is that these attacks do not necessarily have to be carried out by (extremely) powerful intruders with almost unbounded resources. Attacks such as slow DoS attacks [7], [8], [29], [36], [37], [43], [44], asymmetric DoS attacks [34], [46], and amplification attacks [42], [45], can all be carried out by intruders using limited resources. For example, web-servers, such as Apache [43] and NGinx [44] can be successfully attacked by intruders with limited resources using, for instance, mobile phones (SlowDroid [6]).

However, it is hard to determine whether a service is vulnerable to such attacks. While a very powerful intruder with unbounded resources would simply flood the service rendering it unavailable, a resource-bounded intruder carries out an attack by exploiting the protocols used by the target service. He not only triggers a particular sequence of events to consume the service's resources, but also cleverly tries to minimize his effort by triggering events as lazily as possible, renewing service timeouts as late as possible, or by enlisting the help of other benign nodes in the network. Indeed, in many attacks [6], [43], the volume of traffic generated by the intruder is comparable to the volume of a legitimate client thus making it hard for network administrators to even identify when the service is under attack. Therefore, determining such vulnerabilities in advance may help prevent attacks by installing suitable countermeasures.

Intruders can also exploit a wide range of types of resources. For example, Slowloris consumes the limited number of workers web-servers possess; Software Defined Network (SDN) TCAM exhaustion attacks [36] consume the limited amount of TCAM memory of switches; TLS renegotiation DoS attacks [21] consume the server's processing power; SIP forking amplification attacks on Voice-over-IP (VoIP) systems [20], [40], [42], [45] consume the network bandwidth.

While security issues relevant for DoS attacks have been identified, *e.g.*, in [32], where a taxonomy of DoS attacks is given, our main contribution is on formal verification of DDoS. More specifically, our main contributions are the following:

1) We formally define the *DoS problem*. In contrast to existing definitions, such as, the one proposed in [31], our DoS problem takes into account timing aspects, *i.e.*, duration of the

attack. All contributions listed below build on this conceptual advance. *The key technical challenge* is to formally specify behaviors, formalized as traces, where services behave as expected, *e.g.*, triggering timeouts whenever applicable. This is accomplished by specifying configurations that are not allowed, called *critical configurations*. As we use dense time domains, the notion of a trace that does not involve critical configurations, called *non-critical traces*, becomes much more elaborate than when using discrete domains, as one cannot list all moments in dense time that the trace covers. Therefore, ensuring that a trace is non-critical requires checking that all possible decompositions of time advancements, and there are *infinitely many*, do not contain critical configurations.

2) We demonstrate the *expressiveness of our model* by modeling a great number of types of attacks [32]. Besides traditional attacks, such as the SYN flooding attack, our model can also express the types of DoS attacks described above (Slow DoS, Asymmetric and Amplification attacks). Additionally, we also show how to model countermeasures based on traffic monitoring, such as, defenses [38] deployed in web-servers to mitigate the Slowloris attack.
3) We introduce *protocol resource theories*, which refine protocol theories of [16] with timeouts and resource usage. We refine existing Timed Multiset Rewriting (MSR) models of [26] with the notion of non-critical traces;
4) We introduce *resource-bounded intruder models*, which refine the DY intruder with resource usage and action duration. For formal verification, the traditional DY intruder is too powerful for our purposes here, as he can trivially deny any service by, for example, blocking all communication. Inspired by the work [31], we propose, instead, a parametric resource-bounded intruder model, where intruder's actions consume resources;
5) We prove that under suitable conditions, the non-critical reachability problem for models with dense time is PSPACE-complete, which non-trivially advances previous reachability problems [26] that did not consider critical configurations. From this general result, we prove that our DoS problem is PSPACE-complete for a wide class of resource-bounded intruders. Moreover, we prove that the DoS problem is undecidable in general;
6) Finally, we *automate the search* for DoS attacks by *implementing* our formal model in the rewrite tool Maude [12]. This implementation is available at [1]. It is able to find known vulnerabilities within seconds.

The paper starts by describing some examples of known DoS attacks in Section II. In Section III, we present modifications of the Timed MSR of [26]. In Section IV we define protocol resource theories and illustrate these theories with some examples. In Section V we introduce resource-bounded intruder models and describe how our model can be used to specify intruders that can only generate bounded traffic or have bounded processing power. Section VI specifies the DoS problem and illustrates this problem by specifying the Slowloris, Slow-TCAM and TLS Renegotiation DoS attacks. In Section VII we study the complexity of the DoS Problem. Section VIII describes our results on automated search for DoS attacks. Finally, in Section IX we conclude by discussing related work and pointing to future work. Appendices contain supporting material, such as some proofs of complexity results. More details can also be found in the Technical Report [23].

II. Motivating Examples

Traditional DoS attacks, known as flooding or brute-force attacks, target the main service resource by generating large amount of traffic. The most known example is the SYN flooding attack. Such attacks are always possible in the presence of powerful intruders, who can send or intercept an unbounded number of messages. Resource-bounded intruders, on the other hand, exploit protocols used by the target service to perform attacks targeting specific service features related to, *e.g.*, protocols or applications used by the service, as detailed in below examples. Hence, to fully capture various types DoS attacks, all relevant resources should be included in the verification model.

We review some existing DoS attacks that can be carried out by intruders with bounded resources. Attacks aim to consume different resources of the target service, such as the service's threads/workers, available memory, processing power, or even the network bandwidth. However, instead of generating a large amount of traffic, intruders exploit the protocol used by the target service to consume its resources in a lazy manner.

Attacks Consuming Workers/Threads. Web-servers, such as Apache and NGinx, and VoIP servers, such as Asterisk, are subject to attacks that consume all the available workers in their pool of threads. Examples of such attacks include Slowloris, RUDY, Slowread, and Coordinated Call attacks.

For example, Slowloris [43] is an attack on (connection-based) web-servers such as Apache. The intruder exploits the fact that when a web-server receives a GET request with an incomplete header, it allocates one of its workers to attend to this new request. However, since the GET request is incomplete, the worker is left idle and waits for a new piece of the request header until a timeout is triggered (typically 40 seconds). If a new piece arrives and the header is complete, the worker answers the request, but if the header continues to be incomplete, the timeout is reset and the worker waits for another piece until the timeout is triggered. To carry out the Slowloris attack, an intruder sends a burst of incomplete GET requests large enough to occupy all workers (typically 300-400 requests). The intruder does not have to send another burst until a timeout gets close, generating, as a result, very little traffic.

Attacks Consuming Memory. Intruders can target the service's memory. Examples include XML-bombs [46] and Second-Order DDoS attacks [34]. As demonstrated recently [36], even sophisticated networks, such as Software Defined Networks (SDN), can be attacked by resource-bounded intruders. In SDN, switches are general devices that use specialized memories called Ternary Content-Addressable Memory (TCAM) for storing routing rules which are installed by a (powerful) SDN controller. Since TCAMs are expensive and require much energy, SDN switches have a limited amount of TCAM memory capable of storing typically at most 5000 rules. This makes SDN switches subject to Slow-TCAM attacks where the intruder consumes a switch's TCAM memory by forcing the installation of sufficiently many rules. Moreover, to avoid having rules removed, the intruder keeps them alive by sending sporadically new packets that trigger the forwarding rules installed in the switch. Indeed, the intruder can render an SDN switch unavailable by sending less than 4 packets per second.

Attacks Consuming Processing Power. Attacks can also target the processing power of servers. An example of such attack is the Transport Layer Security (TLS) renegotiation DoS attack [21]. TLS is a cryptographic protocol widely used in communications over the Internet. A private connection is established between parties by sharing a private symmetric key created during TLS initialization through a handshake using available public keys. Parties can, however, renegotiate the symmetric key. The initialization process, however, requires more processing power from the server (10 times more effort) than from the client. This amplification leads to the TLS renegotiation DoS attack [21]. The attacker can thus consume the processing power of the server denying its service by issuing a large enough number of renegotiation requests.

Attacks Consuming Network Bandwidth. Instead of targeting a specific resource on a designated server, an intruder with limited resources may target the entire network of servers by mounting an *amplification* DoS attack. Through this attack, the intruder floods the network with messages by expending *minimal* initial effort and exploiting the protocol to enlist the help of other servers in the network, causing the number of messages to amplify to an arbitrary large number while still requiring minimal (or no) further work by the intruder.

An example is the well-known amplification vulnerability in the Session Initiation Protocol (SIP) used to set up VoIP calls [20], [40], [42], [45]. SIP uses a network of SIP proxies to help locate VoIP clients and establish sessions. Generally, to set up a call from client A to another client B, A sends a SIP `INVITE` message (addressed to B) to A's domain SIP proxy, which forwards the message to a SIP proxy of the domain of B, which in turn locates the IP address of B and forwards the invite to B. Typically, however, A's invite message goes through multiple SIP proxies in between. Moreover, a SIP proxy may *fork* an invite message and forward it to multiple nodes (SIP proxies and/or users), a feature that, for example, enables a session to be established with one of several users who can act on behalf of B. However, forking of invite messages makes SIP vulnerable to amplification attacks [20], [42], [45]. Ill-configured or compromised SIP proxies may turn a single invite message into an arbitrarily large number of messages (e.g. through forking loops), flooding the entire network and denying service to users.

III. Timed Multiset Rewriting

We briefly review Timed Multiset Rewriting (MSR) with dense time of [26] which is the language we extend to specify resource-bounded intruders and protocols. Assume a finite first-order typed alphabet, Σ, with variables, constants, function and predicate symbols. Terms and facts are constructed as usual (see [19]), by applying symbols with correct type. For instance, if P is a predicate of type $\tau_1 \times \tau_2 \times \cdots \times \tau_n \rightarrow o$, where o is the type for propositions, and $u_1, \ldots, u_n$ are terms of types $\tau_1, \ldots, \tau_n$, respectively, then $P(u_1, \ldots, u_n)$ is a *fact*.

Timestamped facts are used to specify systems that explicitly mention time. Timestamped facts have the form $F@T$, where F is a fact and T is its timestamp, which can be a variable or a *non-negative real number*. A special predicate $Time$ with arity zero represents the global time. A configuration is a multiset of ground timestamped facts, $\{Time@t, F_1@t_1, \ldots, F_n@t_n\}$, with a single occurrence of a $Time$ fact. We often say facts instead of timestamped facts, for simplicity.

Actions are multiset rewrite rules and are either Time Advancement action or Instantaneous Actions. The action representing the advancement of time, called $Tick$, has the form:

$$Time@T \longrightarrow Time@(T+\varepsilon) \qquad (1)$$

where ε can be instantiated by any positive real number. This specifies that the global time of a configuration can advance by any positive value. Applying the time advancement rule to a given configuration $\{Time@t, F_1@t_1, \ldots, F_n@t_n\}$ yields the configuration $\{Time@(t+\varepsilon), F_1@t_1, \ldots, F_n@t_n\}$ where time advances by ε. We also write $Tick_\varepsilon$ when we refer to the $Tick$ rule (1) for a specific ε.

The remaining actions are instantaneous actions, which do not affect the global time, but may rewrite the remaining facts. They have the following shape:

$$\begin{array}{l} Time@T, W_1@T_1, \ldots, W_k@T_k, F_1@T_1', \ldots, F_n@T_n' \mid \mathcal{C} \longrightarrow \\ \quad \exists \vec{X}.[Time@T, W_1@T_1, \ldots, W_k@T_k, Q_1@(T+D_1), \ldots \\ \qquad \ldots, Q_m@(T+D_m)] \end{array}$$

where $D_1, \ldots, D_m$ are natural numbers and $\mathcal{C}$ is the guard of the action which is a set of constraints involving the time variables appearing in the pre-condition, *i.e.* the variables $T, T_1, \ldots, T_k, T_1', \ldots, T_n'$. Facts $W_1@T_1, \ldots, W_k@T_k$ are preserved by the rule, while $F_1@T_1', \ldots, F_n@T_n'$ are replaced by $Q_1@(T+D_1), \ldots, Q_m@(T+D_m)$.[1] Finally, all free variables appearing in the post-condition must appear in the pre-condition. Constraints are of the form:

$$T \geq T' \pm D, \quad T > T' \pm D \quad \text{and} \quad T = T' \pm D$$

where T and T' are time variables, and D is a natural number, even though time is dense. In the above rules we omit the time constraints whenever the set $\mathcal{C}$ of time constraints is empty.

[1] It has been shown in [26], [27] that relaxing any of the main conditions on instantaneous rules leads to the undecidability of the reachability problem. For example, undecidability is obtained in systems with time constraints that involve three or more time variables. Furthermore, constants Ds and D_is that appear in time constraints and timestamps of created facts are restricted only to natural numbers for computational complexity issues. These values could be generalized to rationals. In fact, for our results it suffces to assume that all these numerical constants mentioned within the above constraints and timestamps are commensurable.

An instantaneous rule of the form $\mathcal{P} \mid \mathcal{C} \longrightarrow \exists \vec{X}.\mathcal{P}'$ can be applied to a configuration $\mathcal{S}$ if there is a subset $\mathcal{S}_0 \subseteq \mathcal{S}$ and a matching substitution θ, such that $\mathcal{S}_0 = \mathcal{P}\theta$ and $\mathcal{C}\theta$ evaluates to true. The configuration resulting from the application of this rule is $(\mathcal{S} \setminus \mathcal{S}_0) \cup ((\mathcal{P}'\sigma)\theta)$, where σ is a substitution that maps the existentially quantified variables $\vec{X}$ to fresh constants, that is, constants not appearing in $\mathcal{S}$. These fresh values are also called *nonces* in protocol security literature [10], [16].[2]

A trace of timed MSR rules $\mathcal{R}$ from a given initial configuration $\mathcal{S}_0$ is a sequence of configurations $\mathcal{S}_0 \longrightarrow_{r_1} \mathcal{S}_1 \longrightarrow_{r_2} \cdots \longrightarrow_{r_n} \mathcal{S}_n$ starting from $\mathcal{S}_0$, such that for all $0 \leq i \leq n-1$, $\mathcal{S}_{i+1}$ is a configuration obtained by applying $r_{i+1} \in \mathcal{R}$ to $\mathcal{S}_i$.

A. Goal and Critical Configurations, Non-Critical Traces

We are not interested in all possible traces, but only traces that do not contain critical configurations and reach some goal. A critical configuration (resp. goal configuration) is specified by a *critical configuration specification* $\mathcal{CS}$ (resp. *goal* $\mathcal{GS}$) which is a set of pairs $\{ \langle \mathcal{S}_1, \mathcal{C}_1 \rangle, \ldots, \langle \mathcal{S}_n, \mathcal{C}_n \rangle \}$. Each pair $\langle \mathcal{S}_j, \mathcal{C}_j \rangle$ is of the form: $\langle \{F_1@T_1, \ldots, F_p@T_p\}, \mathcal{C}_j \rangle$, where $T_1, \ldots, T_p$ are time variables, $F_1, \ldots, F_p$ are facts and $\mathcal{C}_j$ is a set of time constraints involving only variables $T_1, \ldots, T_p$.

We say that a configuration $\mathcal{S}$ is a *critical configuration* w. r. t. $\mathcal{CS}$ (resp. a *goal configuration* w.r.t. $\mathcal{GS}$) if for some $1 \leq i \leq n$, there is a grounding substitution, σ, such that $\mathcal{S}_i\sigma \subseteq \mathcal{S}$ and $\mathcal{C}_i\sigma$ evaluates to true. For example, the configuration $\{Time@3.5,\ F@3.5,\ G@0.2\}$ is critical w.r.t. the critical configuration specification $\{ \langle \{Time@T,\ F@T_1\}, \{T_1 \geq T\} \rangle \}$.

The task of security verification is to check whether a system is vulnerable to an attack. Thus, as we specify in Section VI, a goal configuration will denote that the service has suffered a DoS attack. As we describe in Section IV, the purpose of critical configurations will be to avoid traces where the service does not behave as expected. This is accomplished by only considering non-critical traces defined below.

In discrete time setting [27], a trace was considered critical if it contained a critical configuration. This is no longer adequate when using dense time. To illustrate this, consider, for example, a trace in a timed MSR with dense time, containing the following configurations and a $Tick$:

$$Time@1.5,\ F@3.5 \ \longrightarrow_{Tick_3} \ Time@4.5,\ F@3.5$$

which could potentially be considered as non-critcal w.r.t. the critical configuration specification: $\{\langle\{Time@T,\ F@T_1\},\ \{T_1 = T\}\rangle\}$, as it does not contain any critical configurations. However, a trace containing rules $Tick_1$ and $Tick_2$: $Time@1.5,\ F@3.5 \longrightarrow_{Tick_2} Time@3.5,\ F@3.5 \longrightarrow_{Tick_1} Time@4.5,\ F@3.5$, would be critical w.r.t. the same critical configuration specification since it contains the critical configuration $\{Time@3.5,\ F@3.5\}$. Clearly this is not appropriate. A configuration that is not critical may turn into a critical configuration purely because the time is ticking. While in discrete time setting [27] one could list all (discrete time) moments in a time window, this is not possible when time is dense. The definition of non-critical traces in dense time setting is necessarily more involved.

Definition III.1 (Non-Critical Traces)**.** *Let $\mathcal{R}$ be a set of timed MSR rules and $\mathcal{CS}$ be a critical configuration specification. A trace $\mathcal{P}$ of $\mathcal{R}$ rules is* non-critical *if it contains no critical configuration and if no critical configuration is reached along any trace obtained by matching any subtrace of $\mathcal{P}$ on the left below with the one on the right:*

$$\mathcal{S}_i \longrightarrow_{Tick_\varepsilon} \mathcal{S}_{i+1} \qquad\qquad \mathcal{S}_i \longrightarrow_{Tick_{\varepsilon_1}} \mathcal{S}' \longrightarrow_{Tick_{\varepsilon_2}} \mathcal{S}_{i+1}$$

where ε_1 and ε_2 are arbitrary non-negative real numbers, such that, $\varepsilon = \varepsilon_1 + \varepsilon_2$ holds.

That is, one has to consider all possible ways to decompose $Tick$s in a trace. This ensures that the continuity of time and the notion of non-critical traces are well combined. Checking whether a given trace in a system with dense time is non-critical is potentially more challenging than in the untimed setting [28] and models with discrete time [27] as it requires potentially checking through an infinite number of traces. This could affect the complexity of the corresponding verification problem. Fortunately, as presented in Section VII, by using abstract representations, we are able to deal with these challenges in an elegant manner that does not impact the problem complexity.

Balanced rules were introduced in [41]. Systems containing only balanced rules represent an important class of systems, *balanced systems*, for which several reachability problems have been shown decidable [24]–[26]. A rule is balanced if the number of facts appearing in its precondition and in its post-condition is the same. As described in [25], any unbalanced rule can be made balanced by using dummy facts. For example, the unbalanced rule: $Time@T, F_1@T_1 \longrightarrow Time@T, F_1@T_1, F_2@T_2$ can be turned into a balanced rule by adding a dummy fact to its pre-condition, $Time@T, F_1@T_1, P@T_3 \longrightarrow Time@T, F_1@T_1, F_2@T_2$. Here, we are going to consider two different types of dummy facts, facts $P(I)$ for intruders, and facts $D(s)$ for services.

There are important implications when using balanced systems, *e.g.*, balanced protocol specifications and balanced intruder models. The number of parallel protocol sessions running at the same time is bounded, although the total number of protocol sessions is unbounded. Moreover, the number of facts (and thus symbols) that the balanced intruder can remember is bounded. As shown in [25], however, many of the usual known attacks on protocols can be carried out by balanced intruders.

Such balanced systems have the following important property [41]:

Proposition III.2. *Let $\mathcal{R}$ be a set of balanced rules. Let $\mathcal{S}_0$ be a configuration with exactly m facts. Let $\mathcal{S}_0 \longrightarrow \cdots \longrightarrow \mathcal{S}_n$ be an arbitrary trace of rules $\mathcal{R}$ starting from $\mathcal{S}_0$. Then for all $0 \leq i \leq n$, $\mathcal{S}_i$ has exactly m facts.*

For some of our complexity results (in Section VII), we will assume an upper-bound on the size of facts. The size, $|F@t|$, of a timed fact $F@t$ is the number of symbols in F. For example, $|M(a, \{a, b\}_k)@t| = 5$.

[2] Substitution application $(\mathcal{S}\theta)$ is defined as usual [19], *i.e.*, by mapping time variables in $\mathcal{S}$ to non-negative real numbers, nonce names to nonce names (renaming of nonces) and term variables to terms.

B. Signature for Protocols and Resource-Bounded Intruders

For our DoS problem, we will use signatures containing the constants, functions, and predicate symbols described below.

Message Expressions: We assume a message signature Σ of constants and function symbols. Constants include nonces, symmetric keys and player names. Messages are constructed as usual, using constants, variables and at least the following function symbols: $\mathsf{sk}(\mathsf{p})$ for the secret key of the player p, $\mathsf{pk}(\mathsf{p})$ its public key, $\{\mathsf{m}\}_\mathsf{k}$ the encryption of m using key k, and the tuple function $\langle \mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_n \rangle$ denoting a list of messages $\mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_n$. (Other crypto constructs, such as MAC and hash, can be also added as usual.) We define $(\mathsf{pk}(\mathsf{p}))^{-1} = \mathsf{sk}(\mathsf{p})$ and $\mathsf{k}^{-1} = \mathsf{k}$ if k is a symmetric key. We also write interchangeably the singleton tuple $\langle \mathsf{m} \rangle$ and m.

Resources: For simplicity, we will consider resources as natural numbers. We could also use more abstract definitions as the ones used in [31], *e.g.*, monoids. The results in the paper are not affected by this change. Each service s is associated with a *minimal service resource* value, r^s_m, specifying the lower bound on resources required by the service to work. For example, for VoIP servers this would typically be 0 workers. Once resources reach r^s_m, the service is considered exhausted. We will also assume an *initial service resource* for a service s, written r^s_{ini}.

Predicates: Besides the global time predicate $Time$ described above, our signature contains the following predicate symbols. The fact $\mathsf{N}(\mathsf{m})@t$ denotes that message m is on the network, available for receipt from the moment t. Furthermore, the facts $\mathsf{S}_i(id, s_{id}, r, \vec{x}_i)$, where $0 \leq i \leq n$, specify $n+1$ protocol states for service id. The number of states and their arguments, $\vec{x}_i$, are protocol-specific. s_{id} is a protocol session identification symbol, which is unique per protocol session, r_i is a natural number specifying the number of resources allocated by the service to maintain the protocol in the state S_i, and the timestamp t of S_i specifies the moment until which the protocol is allowed to be in state S_i, *i.e.*, the timeout of that protocol state. In addition, $\mathsf{R}(id, r)@t$, denotes that service/intruder id has r resources available at moment t; $\mathsf{Av}(id)@t$ and $\mathsf{Den}(id)@t$, denote, respectively, that the service id is available or unavailable from moment t; $\mathsf{M}(id, \mathsf{m})@t$, denotes that the intruder id knows the message m at moment t; and $\mathsf{Rec}(id, r)@t$, denotes that the intruder id can recover r resources at moment t.

IV. Protocol Specifications

We introduce a language for formal protocol specification of how resources are consumed during protocol execution and the timeouts, which specify when protocols may be terminated and resources recovered. We do so by extending protocol theories proposed in [16], [25].

For readability and simplification of exposition of our model and the related complexity results, our protocol and intruder specifications contain exactly one resource. Generalization to multiple resources is straightforward, by including specific resource predicates for each of the relevant resources, such as service's threads/workers, CPU time, network bandwidth, etc.

Definition IV.1 (Protocol Resource Theory). *A protocol resource theory of a service s is specified by a set of state predicates $\mathsf{S}_0, \mathsf{S}_1, \ldots, \mathsf{S}_n$, its minimal resource r^s_m, and the rules of the following form:*

- ***protocol initialization rule:***

$$\begin{array}{l} Time@T, \mathsf{R}(s, r_I + R + r^s_m)@T_1, \mathsf{N}(\mathsf{m}_I)@T_2 \\ \quad \mid\ T_1 \leq T, T_2 \leq T \longrightarrow \\ \exists S_{id}.[Time@T, \mathsf{S}_0(s, S_{id}, r_I, \vec{x})@(T + t_I), \\ \qquad \mathsf{R}(s, R + r^s_m)@T, \mathcal{W}_1] \end{array}$$

  where S_0 is the initial state, r_I is a natural number specifying the initialization cost for the service, t_I is a natural number specifying the timeout of the initial service state, m_I is an initialization message, S_{id} is a fresh protocol session identification token and $\mathcal{W}_1$ is a multiset of facts;

- ***protocol execution rules:***

$$\begin{array}{l} Time@T, \mathsf{S}_i(s, S_{id}, r_i, \vec{x}_i)@T_1, \\ \quad \mathsf{R}(s, r_j - r_i + R + r^s_m)@T_2, \mathsf{N}(\mathsf{m}_i)@T_3, \mathcal{W}_1, \mathcal{W} \\ \qquad \mid\ T_1 \geq T, T_2 \leq T, T_3 \leq T \longrightarrow \\ Time@T, \mathsf{S}_j(s, S_{id}, r_j, \vec{x}_j)@(T + t_j), \\ \quad \mathsf{R}(s, R + r^s_m)@T, \mathcal{W}_2, \mathcal{W} \end{array}$$

  where for $0 \leq i \leq n$, r_i and r_j are natural numbers, specifying the execution costs associated to state S_i and S_j, respectively, t_j is the timeout of the service for state S_j, m_i is a (state transition) message, and $\mathcal{W}, \mathcal{W}_1, \mathcal{W}_2$ are arbitrary multisets of facts. We also assume here that $r_j - r_i + R > 0$, that is, the service has enough resources;

- ***protocol state timeout rule:*** *where $0 \leq i \leq n$:*

$$\begin{array}{l} Time@T, \mathsf{R}(s, R)@T_1, \mathsf{S}_i(s, S_{id}, r_i, \vec{x}_i)@T \quad \longrightarrow \\ \quad Time@T, \mathsf{R}(s, r_i + R)@T \end{array}$$

- ***service availability rules:***

$$\begin{array}{l} Time@T,\ \ \mathsf{R}(s, r^s_m)@T, \mathsf{Av}(s)@T_2 \ \longrightarrow \\ \quad Time@T,\ \ \mathsf{R}(s, r^s_m)@T, \mathsf{Den}(s)@T \\ Time@T,\ \ \mathsf{R}(s, r^s_m + R + 1)@T, \mathsf{Den}(s)@T_2 \longrightarrow \\ \quad Time@T,\ \ \mathsf{R}(s, r^s_m + R + 1)@T, \mathsf{Av}(s)@T \end{array}$$

Intuitively, a service may run a number of protocol sessions in parallel. In order to maintain each protocol session, the service may need to allocate resources, such as memory, CPU time, workers, etc, for some time. In particular, the protocol initialization rule specifies that the creation of a new protocol session requires r_I resources and these may be recovered t_I time units later. Notice the use of the existential quantifier that creates a fresh protocol session identifier. Protocol execution rules change the state of the protocol session from S_i to S_j, updating the resources allocated and the timeout. At the same time, validity of the current protocol state, *i.e.*, protocol state timeouts are checked through constraints involving timestamps of protocol states and the global time T, $T_1 \geq T$. The protocol timeout rule specifies that a protocol session is forgotten when the timeout is reached. Service availability rules simply specify that a service is denied when resources reach a minimum, and available if the resources are greater than the minimum.

Protocol Critical Configuration Specification As rules can be applied in a non-deterministic fashion, undesirable traces where a service misbehaves are allowed. For example, it is possible to construct traces where the protocol state timeout rule is never applied, and thus the service never releases resources allocated to protocol sessions that should have been ended by a timeout. We will classify such traces where the service does not behave as expected as critical. This is formally achieved by protocol critical configuration specifications defined below:

Definition IV.2 (Protocol $\mathcal{CS}$)**.** *The* protocol critical configuration specifications *(protocol $\mathcal{CS}$) for a given protocol theory of a service s, with minimal resource r^s_m and with protocol state predicates S_i, with $0 \le i \le n$, is composed of the three types of critical configuration specifications:*

- ***Timeout** $\mathcal{CS}$: for all* $0 \le i \le n$
  $\langle\{Time@T, S_i(s, S_{id}, R_i, \vec{x}_i)@T_1\}, \{T_1 < T\}\rangle$;
- ***Denied** $\mathcal{CS}$:* $\langle\{\mathsf{R}(s, r^s_m)@T_1, \mathsf{Av}(s)@T_2\}, \{T_1 \le T_2\}\rangle$;
- ***Available** $\mathcal{CS}$:*
  $\langle\{\mathsf{R}(s, r^s_m + R + 1)@T_1, \mathsf{Den}(s)@T_2\}, \{T_1 \le T_2\}\rangle$.

Timeout $\mathcal{CS}$ specifies that configurations denoting protocol sessions for which the timeout has passed are critical. Denied $\mathcal{CS}$ specifies that configurations are critical if a service is considered available at a time its resources have been exhausted. Similarly, as per Available $\mathcal{CS}$, service should not be considered denied at anytime when sufficient resources are available.

Definition IV.3 (Service)**.** *A service $\mathcal{A}$ has the following components:*

- *A unique identification symbol s;*
- *A minimal service resource value r^s_m;*
- *An initial service resource value r^s_{ini}, such that $r^s_{ini} > r^s_m$;*
- *A set of protocol theories involving only s and their corresponding protocol $\mathcal{CS}$. We assume that the set of protocol state predicates of different protocol theories are disjoint.*

Additionally, we say that a service is balanced, if all protocol theories are balanced. This is obtained using the dummy facts $D(s)$ (see Section III). Moreover, a balanced service also contains a natural number d_s specifying the number of $D(s)$ facts available.

Intuitively, balanced services can only maintain a bounded number of parallel protocol sessions, since for each protocol session a D fact is consumed and there is a bounded number, d_s, of D facts available.

A. Examples of Protocol Theories

We now describe four examples of protocol theories: GET-HTTP, SDN rule insertion, TLS renegotiation and SIP forking. While the GET-HTTP example is given in some detail along with its protocol theory specification, the other three examples are only briefly described for brevity. They can be modeled similarly to the GET-HTTP protocol, but by using different types of resources and states.

GET-HTTP: We specify (an abstract version of) the HTTP GET method, which is subject to the Slowloris attack [43], exhausting the web-server's workers, as described in Section II. The initialization and execution rules are depicted in Fig. 1.

$$\begin{array}{ll}
\textbf{INIT:} & Time@T,\ \mathsf{R}(s, 1 + R + r^s_m)@T_1,\ N(INIT)@T_2 \\
& \quad \mid\ T_1 \le T,\ T_2 \le T \longrightarrow \\
& \exists S_{id}.[Time@T, \mathsf{S}_0(s, S_{id}, 1)@(T + 40), \mathsf{R}(s, R + r^s_m)@T] \\
\textbf{GET:} & Time@T,\ S_0(s, S_{id}, 1)@T_1,\ \mathsf{R}(s, R)@T_2,\ N(GET)@T_3 \\
& \quad \mid\ T_1 \ge T,\ T_2 \le T,\ T_3 \le T \longrightarrow \\
& \quad Time@T,\ S_1(s, S_{id}, 1)@(T + 40),\ \mathsf{R}(s, R)@T \\
\textbf{INC:} & Time@T,\ S_1(s, S_{id}, 1)@T_1,\ \mathsf{R}(s, R)@T_2,\ N(Inc)@T_3 \\
& \quad \mid\ T_1 \ge T,\ T_2 \le T,\ T_3 \le T\ \longrightarrow \\
& \quad Time@T,\ S_1(s, S_{id}, 1)@(T + 40),\ \mathsf{R}(s, R)@T \\
\textbf{COM1:} & Time@T,\ S_0(s, S_{id}, 1)@T_1,\ \mathsf{R}(s, R)@T_2,\ N(Com)@T_3 \\
& \quad \mid\ T_1 \ge T,\ T_2 \le T,\ T_3 \le T\ \longrightarrow \\
& \quad Time@T,\ S_2(s, S_{id}, 1)@T,\ \mathsf{R}(s, R)@T \\
\textbf{COM2:} & Time@T,\ S_1(s, S_{id}, 1)@T_1,\ \mathsf{R}(s, R)@T_2,\ N(Com)@T_3 \\
& \quad \mid\ T_1 \ge T,\ T_2 \le T,\ T_3 \le T\ \longrightarrow \\
& \quad Time@T,\ S_2(s, S_{id}, 1)@T,\ \mathsf{R}(s, R)@T
\end{array}$$

Fig. 1. Protocol Resource Theory for HTTP GET Protocol Method. We elide the protocol state timeout rules and service availability rules.

The protocol has three states, $\mathsf{S}_0, \mathsf{S}_1$ and S_2, which, respectively, correspond to the state when the HTTP protocol session is initialized (S_0), *i.e.*, by performing the SYN-ACK which is omitted for brevity, the state where a incomplete GET request is received (S_1), and the state where the GET request is completed and the protocol session may end (S_2).

Each protocol state requires one resource, corresponding to one worker as is the case with connection based web-servers, such as Apache. The timeout of each protocol state is 40 time units, that is, if no further interaction is performed within 40 time units then the protocol session is terminated.

The rule INIT specifies the protocol initialization, where one worker is allocated and a fresh protocol session S_{id} is created. The rules GET, INC, COM1 and COM2 specify the transitions between the states as described above. Notice that in all states the allocated worker is kept allocated. Here we use Inc to represent a message which has not completed the GET header and Com for a message that has completed the header. In practice, a message header is complete if it ends with \r \n \r \n and incomplete otherwise. Thus, it is possible to move from state S_0 to the final state S_2 (rule COM1) by sending a complete message or from S_1 to S_2 by first sending an incomplete message (rule INC) and then a complete message (rule COM2).

SDN Rule Addition: Software Defined Networks (SDN) use protocols, such as OpenFlow [35], to install rules in SDN switches. We use as resources the number of available rules that can be stored in an SDN Switch, typically at most 5000-8000 rules. Whenever an SDN switch receives a packet for which there is no applicable SDN forwarding rule, it installs a new rule after receiving rule installation approval from the centralized SDN controller. This is modeled by a protocol initialization rule. Rules come with a timeout. If no further packets arrive that activate this rule before the timeout, then the forwarding rule is dropped. This can be modeled with the protocol state timeout rule. Otherwise, if a packet is received, then the timeout is reset. This can be modeled using a protocol execution rule.

TLS Renegotiation: One can also model the Transport Layer Security (TLS) renegotiation protocol using the number of handshakes being processed as the resource. For example, according to [21] a server can handle between 150-300 handshakes per second. This is proportional to the CPU power of the server. Whenever a new renegotiation is requested, the server consumes processing power. TLS protocols also illustrate the use of messages involving cryptographic operators for DoS attacks. The use of asymmetric keys in handshakes and the creation of fresh keys causes overheads to the server.

SIP invite forking: A protocol theory of SIP specifies the mechanism of forwarding and forking invite messages, which is known to be vulnerable to amplification attacks [42], [45]. As amplification targets the network's bandwidth, we identify the service s as the network and the resource r as the network's bandwidth (measured in terms of invite messages sent). When an invite message is first introduced, a session is created. A state in the protocol maintains the total number of forwarded or forked invite messages for a session, along with the session's timeout. While both forwarding and forking messages reset the timeout of their corresponding sessions, only forking increases the amount of resources allocated for the session.

B. Modeling Time-Based Countermeasures

We now illustrate how we can extend our model to take into account countermeasures (CMs) for DDoS attacks based on timeouts, such as, ReqTimeOut [38] for mitigating the Slowloris attack. The basic idea of timeout based CM is to trigger a timeout whenever some condition on the traffic is satisfied, thus forcing that a connection is closed and the service's resources are made available.

For example, for SYN flooding attacks, timeout based CMs monitor whether a SYN-ACK handshake is completed within some specified timeout. If a timeout is triggered, then the connection is closed and the resources allocated by this connection can be used by another connection. Similarly, ReqTimeOut is a time based CM which monitors whether a packet header or packet body is completed within some specified time. If a timeout is triggered, then the connection is closed and the allocated worker can be used to serve another connection. It is, therefore, used to mitigate attacks such as Slowloris which take too long to complete the header.

To formalize timeout based CM, we use the predicate:

- $\mathsf{TimeCM}(s, S_{id})@T$, denoting that the CM triggers a timeout at time T which closes the connection S_{id} of service s.

This is specified by the following countermeasure timeout rule:

$$\begin{array}{l} \mathit{Time}@T, \mathsf{R}(s, R)@T_1, \mathsf{S}_i(s, S_{id}, r_i, \vec{x}_i)@T_2, \\ \quad \mathsf{TimeCM}(s, S_{id})@T \longrightarrow \mathit{Time}@T, \mathsf{R}(s, r_i + R)@T \end{array}$$

The protocol initialization and execution rules may update TimeCM (contained in $\mathcal{W}_1$ in Definition IV.1). We specify, additionally, the following critical configuration for CM: $\langle\{\mathit{Time}@T, \mathsf{TimeCM}(s, S_{id})@T_1\}, \{T_1 < T\}\rangle$

This is similar to the timeout $\mathcal{CS}$ in Definition IV.2, as we only consider traces where CMs trigger timeouts.

ReqTimeOut: Using the above machinery, we specify the ReqTimeOut [38], a time-based CM available in the Apache Web-Server for mitigating Slow-DoS attacks. One specifies two natural numbers: **Header Timeout** (h_{TO}) denoting the maximum elapsed time for the connection to send the complete header; **Body Timeout** (b_{TO}) denoting the maximum elapsed time for the connection to send the complete packet body.

Thus, a protocol with ReqTimeOut has three states: S_{HI}, denoting a protocol state where the connection did not send a complete header; S_{HC-BI}, denoting a protocol state where the connection sent a complete header, but not the complete packet body; S_{BC}, denoting a protocol state where the connection sent a complete header and body.

The following protocol initialization rule sets the TimeCM:

$$\begin{array}{l} \mathit{Time}@T,\ \mathsf{R}(s, 1 + R + r^s_m)@T_1,\ N(\mathit{INIT})@T_2 \\ \quad \mid\ T_1 \leq T,\ T_2 \leq T \longrightarrow \\ \exists S_{id}.[\mathit{Time}@T, \mathsf{S}_{HI}(s, S_{id}, 1)@(T + 40), \mathsf{R}(s, R + r^s_m)@T, \\ \quad \mathsf{TimeCM}(s, S_{id})@(T + h_{TO})] \end{array}$$

The protocol execution rule specifies when the header is completed then the timeout is set for receiving the packet body:

$$\begin{array}{l} \mathit{Time}@T,\ \mathsf{S}_{HI}(s, S_{id}, 1)@T_1,\ \mathsf{R}(s, R)@T_2,\ N(M)@T_3, \\ \quad \mathsf{TimeCM}(s, S_{id})@T_4\ \mid\ T_1 \geq T,\ T_2 \leq T,\ T_3 \leq T \longrightarrow \\ \mathit{Time}@T,\ \mathsf{S}_{HC-BI}(s, S_{id}, 1)@(T + 40),\ \mathsf{R}(s, R)@T \\ \quad \mathsf{TimeCM}(s, S_{id})@(T + b_{TO}) \end{array}$$

A similar rule specifies when the body of the packet is completed. We elide this rule.

Discussion It should be possible to model more refined time-based CMs by adding suitable facts and rules. For example, adding a predicate that remembers the number of bytes received can be used to model more advanced configurations of the ReqTimeOut, which is activated depending on the traffic rate.

V. Resource-Bounded Intruder Model

This Section introduces a novel parametric intruder model that is based on the powerful Dolev-Yao (DY) intruder [15], but has bounded resources. In contrast to the DY intruder, the resource-bounded intruder can only consume a bounded number of his resources in any given time window.

Provided he has enough resources, the resource-bounded intruder can compose, decompose, encrypt and decrypt messages for which he knows the appropriate key, and generate fresh values. The rules corresponding to these actions, depicted in Fig. 2, are based on the DY intruder rules [16], but refined with the notion of time as in Timed DY intruders [33] and with resource consumption. Each rule has an associated cost, specified by a SPEC function, returning a triple of natural numbers $\langle\delta_L, \delta_R, r_R\rangle$: δ_L is the time for carrying out the action; r_R denotes resources consumed by the action, which can only be re-used after δ_R time units. We assume all SPEC functions are computable in polynomial time.

I/O Rules:

REC:
$$\begin{aligned}&Time@T, N(X)@T_1, \mathsf{R}(I, Z + r_R)@T_2 \mid T \geq T_1 \longrightarrow\\&\quad Time@T, M(I,X)@(T+\delta_L), \mathsf{R}(I,Z)@T,\\&\qquad \mathsf{Rec}(I, r_R)@(T+\delta_R)\\&\text{where } \textsc{spec}_{REC}(X,I) = \langle \delta_L, \delta_R, r_R \rangle\end{aligned}$$

SND:
$$\begin{aligned}&Time@T, M(I,X)@T_1, \mathsf{R}(I, Z + r_R)@T_2 \mid T \geq T_1 \longrightarrow\\&\quad Time@T, N(X)@(T+\delta_L), \mathsf{R}(I,Z)@T,\\&\qquad \mathsf{Rec}(I, r_R)@(T+\delta_R)\\&\text{where } \textsc{spec}_{SND}(X,I) = \langle \delta_L, \delta_R, r_R \rangle\end{aligned}$$

Message Composition and Decomposition Rules:

CMP:
$$\begin{aligned}&Time@T, M(I,X)@T_1, M(I,Y)@T_2, \mathsf{R}(I, Z + r_R)@T_3\\&\longrightarrow Time@T, M(I, \langle X, Y \rangle)@(T+\delta_L), \mathsf{R}(I,Z)@T,\\&\qquad \mathsf{Rec}(I, r_R)@(T+\delta_R)\\&\text{where } \textsc{spec}_{COMP}(X,Y,I) = \langle \delta_L, \delta_R, r_R \rangle\end{aligned}$$

DCM:
$$\begin{aligned}&Time@T, M(I, \langle X, Y \rangle)@T_1, \mathsf{R}(I, Z + r_R)@T_2 \longrightarrow\\&\quad Time@T, M(I,X)@(T+\delta_L), M(I,Y)@(T+\delta_L),\\&\qquad \mathsf{R}(I,Z)@T, \mathsf{Rec}(I, r_R)@(T+\delta_R)\\&\text{where } \textsc{spec}_{DCMP}(\langle X, Y \rangle, I) = \langle \delta_L, \delta_R, r_R \rangle\end{aligned}$$

USE:
$$\begin{aligned}&Time@T, M(I,X)@T_1, \mathsf{R}(I, Z + r_R)@T_2 \longrightarrow\\&\quad Time@T, M(I,X)@T_1, M(I,X)@(T+\delta_L),\\&\qquad \mathsf{R}(I,Z)@T, \mathsf{Rec}(I, r_R)@(T+\delta_R),\\&\text{where } \textsc{spec}_{USE}(X,I) = \langle \delta_L, \delta_R, r_R \rangle\end{aligned}$$

ENC:
$$\begin{aligned}&Time@T, M(I,K)@T_1, M(I,X)@T_2, \mathsf{R}(I, Z + r_R)@T_3\\&\longrightarrow Time@T, M(I, \{X\}_K)@(T+\delta_L), M(I,K)@T_1,\\&\quad M(I,X)@T_2, \mathsf{R}(I,Z)@T, \mathsf{Rec}@(I, r_R)(T+\delta_L)\\&\text{where } \textsc{spec}_{ENC}(K,X,I) = \langle \delta_L, \delta_R, r_R \rangle\end{aligned}$$

DEC:
$$\begin{aligned}&Time@T, M(I,K^{-1})@T_1, M(I, \{X\}_K)@T_2,\\&\qquad \mathsf{R}(I, Z + r_R)@T_3 \longrightarrow\\&\quad Time@T, M(I,X)@(T+\delta_L), M(I,K^{-1})@T_1,\\&\qquad M(I, \{X\}_K)@T_2, \mathsf{R}(I,Z)@T, \mathsf{Rec}(I, r_R)@(T+\delta_R)\\&\text{where } \textsc{spec}_{DEC}(K^{-1}, \{X\}_K, I) = \langle \delta_L, \delta_R, r_R \rangle\end{aligned}$$

GEN:
$$\begin{aligned}&Time@T, \mathsf{R}(I, Z + r_R)@T_1 \longrightarrow\\&\quad \exists N. Time@T, M(I,N)@(T+\delta_L), \mathsf{R}(I,Z)@T,\\&\qquad \mathsf{Rec}(I, r_R)@(T+\delta_R)\\&\text{where } \textsc{spec}_{GEN}(I) = \langle \delta_L, \delta_R, r_R \rangle\end{aligned}$$

Resource Maintenance Rule:

RES:
$$\begin{aligned}&Time@T, \mathsf{R}(I,Z)@T_1, \mathsf{Rec}(I, r_R)@T_2 \mid T_2 \leq T \longrightarrow\\&\quad Time@T, \mathsf{R}(I, Z + r_R)@T\end{aligned}$$

Fig. 2. Bounded Resource Intruder Theory $\mathcal{I}$

As with protocol theories introduced in Section IV, intruder resources may represent *e.g.*, traffic generation or CPU consumption. Again, for simplicity of exposition, we consider resources to be represented by natural numbers and use only one resource.

Bounded resource intruder rules, given in Fig. 2, denote the following intruder actions: **REC** rule specifies the intruder action of receiving a message from the network. The rule's cost is specified by $\textsc{spec}_{REC}(X,I) = \langle \delta_L, \delta_R, r_R \rangle$: the intruder I consumes r_R resources for receiving the message X, taking δ_L time units to learn X;[3] **SND** rule specifies sending of a message to the network with the cost $\textsc{spec}_{SND}(X,I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for sending the message X, taking δ_L time units; **CMP** rule specifies composing two messages known to the intruder, costing $\textsc{spec}_{COMP}(X,Y,I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for composing messages X and Y into message $\langle X, Y \rangle$ in δ_L time units; **DCM** rule specifies decomposing, costing $\textsc{spec}_{DCMP}(X,Y,I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for decomposing $\langle X, Y \rangle$ into X and Y in δ_L time units; **USE** rule specifies the copying of a known message, costing $\textsc{spec}_{USE}(X,I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for copying X in δ_L time units; **ENC** rule specifies encryption, having the cost $\textsc{spec}_{ENC}(K,M,I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for encrypting M using the key K, taking δ_L time units; **DEC** rule specifies decryption, costing $\textsc{spec}_{DEC}(K^{-1}, M_K, I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources for decrypting M_K using the key K^{-1} and learning the message M, taking δ_L time units; **GEN** rule specifies creating of a fresh value, *e.g.*, a nonce or a fresh key. Its cost is $\textsc{spec}_{GEN}(I) = \langle \delta_L, \delta_R, r_R \rangle$: I consumes r_R resources, taking δ_L time units; **RES** rule denotes recovery of available resources.

Definition V.1 (Resource-Bounded Intruder)**.** *A resource-bounded intruder, $\mathcal{I}$, consists of his unique identification symbol I, his maximal resource r^I_{max}, a finite set $\mathcal{M} = \{M(I, \mathsf{m}_1)@0, \ldots, M(I, \mathsf{m}_n)@0\}$ of M facts specifying intruder's initial knowledge base and definitions of* $\textsc{spec}_R$ *functions, for each rule R given in Fig. 2.*

Types of Resource-Bounded Intruders: Different bounded intruders can be specified by using different definitions of $\textsc{spec}_R$ functions. We illustrate this feature with some examples. Assume that the intruder has at most r^I_{max} resources.

Bounded Traffic Intruder Model represents the intruder that can send only a number of messages at a given rate, *e.g.*, messages per second. This is achieved by specifying $\textsc{spec}_{SND}$ and $\textsc{spec}_{REC}$ accordingly, and setting $\textsc{spec}_R = \langle 0, 0, 0 \rangle$ for the remaining rules. For example, setting $\textsc{spec}_{SND}(X,I) = \textsc{spec}_{REC}(X,I) = \langle 1, 1, 1 \rangle$ for all X means that the intruder can only generate/intercept traffic at a maximum rate of r^I_{max} messages per time unit. This is because sending or receiving a message consumes one of his resources for 1 time unit. Since he has only r^I_{max} resources, he can send at most r^I_{max} messages in a time unit. One could refine this even further by specifying $\textsc{spec}_{SND}(X,I)$ to depend on X, so that, *e.g.*, the number of resources is proportional to the number of symbols of X, so that the intruder is only capable of sending r^I_{max} symbols per time unit. Similarly, one could consider that sending encrypted messages requires more resources than sending plaintext.

Bounded Processing Intruder Model: One can also specify the intruder that can only carry out a bounded number of actions in a given time window according to his processing power. The maximum resources may be expressed in terms of percentage of available CPU, *i.e.*, $r^I_{max} = 100$, or even in the number of CPU cycles for more precise models. Each action would then consume CPU resources for some given time. For example, the cost of encrypting and decrypting, $\textsc{spec}_{ENC}$ and $\textsc{spec}_{DEC}$, will impact the CPU usage depending on the key and message being encrypted or decrypted.

[3] Notice that we do not explicitly consider transmission time here. This can be done by adding explicit delays if the network topology is known. However, these delays are normally in a lower order of magnitude than service timeouts, and they are not very relevant for DoS attacks. This is in contrast to Cyber-Physical Security Protocols where transmission delays are relevant [26], [33].

Bounded Memory Intruder Models: Bounded memory intruder models are not specified by using $\textsc{spec}_R$ functions, but by using balanced intruder models as described in [25]. All the rules of the intruder model are transformed into balanced rules as described in Section III, using dummy facts $P(I)@T$. For example, the balanced version of the REC rule is as follows:

$$\begin{array}{l} Time@T, N(X)@T_1, \mathsf{R}(I, Z + r_R)@T_2, P(I)@T_3 \mid T \geq T_1 \longrightarrow \\ \quad Time@T, M(I,X)@(T+\delta_L), \mathsf{R}(I,Z)@T, \mathsf{Rec}(I, r_R)@(T+\delta_R) \end{array}$$

Notice that a dummy fact is consumed when a message is received. Since the number of dummy facts in a configuration is bounded, the number of messages that can be learned from the network is bounded. Memory bounded intruders should also be able to manage their memory. This is specified by the following memory maintenance rule that enables the intruder to delete information form his memory:

$$\textbf{DELM: } Time@T, M(I,X)@T_1 \longrightarrow Time@T, P(I)@T. \quad (2)$$

All the rules of a balanced bounded resource intruder theory are shown in Fig. 3.

Definition V.2 (Balanced Resource-Bounded Intruder)**.** *A balanced resource-bounded intruder, $\mathcal{I}$, consists of his unique identification symbol I, his maximal resource r^I_{max}, a natural number dum, specifying the number of dummy facts available to the intruder, a finite set of M facts specifying the intruder's initial knowledge base, and definitions of* $\textsc{spec}_R$ *functions, for all balanced versions of the rules R given in Fig. 2 and* **DELM** *rule (2).*

Notice that if for all rules R, $\textsc{spec}_R = \langle 0,0,0 \rangle$, then the (unbalanced) intruder model in Fig. 2 is equivalent to the DY intruder [16]. All actions can be performed at no cost (in time or resources) to the intruder. Indeed, he can generate and intercept any number of messages. Finally, one can also imagine a lattice of intruder models with order defined by the number of resources where, clearly, the DY intruder is the most powerful.

VI. DoS Problem

We now formulate DoS attacks in our framework, and contrast it with notions of DoS attack from existing literature.

In [31] a DoS attack is viewed as "the resource exhaustion attack, in which an attacker, by initiating a large number of instances of a protocol, causes a victim to exhaust his resources". According to [9], a DoS "is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service". As per US Department of Homeland Security official website [14], a DoS attack occurs "when legitimate users are unable to access information systems, devices, or other network resources due to the actions of a malicious cyber threat actor". Our formulation of DoS attacks addresses the issues from the above definitions. The notion of DoS attack from [31] is further refined by an additional duration parameter that we find is relevant for the following reasons. Flooding attacks are always possible in the presence of powerful attackers. However, very short service interruptions may be tolerated in practice, while prolonged unavailability of service would be considered as a successful DoS attack. As the access to server resources is not instant, due to, *e.g.*, network delays, users may not consider a short delay in service as denial of service.

Intuitively, a DoS attack on a service is successful if the service's resources are exhausted for some *duration*, $mdur$. More precisely, the verification task is to determine whether, in the presence of attackers, some service is subject to a DoS attack, by searching for a non-critical trace of the form: $\mathcal{S}_0 \longrightarrow \mathcal{S}_1 \longrightarrow \cdots \longrightarrow \mathcal{S}_i \longrightarrow \cdots \longrightarrow \mathcal{S}_{i+m} \longrightarrow \cdots \longrightarrow \mathcal{S}_n$, where, $\mathcal{S}_0$ is the initial configuration of the verification scenario, the global time, t_i, in the configuration $\mathcal{S}_i$, and the global time, t_{i+m} in the configuration $\mathcal{S}_{i+m}$, are such that $t_{i+m} - t_i \geq mdur$, and that for a service, s, its resources in all configurations between $\mathcal{S}_i$ and $\mathcal{S}_{i+m}$ are less or equal to r^s_m.

Notice that the definition below of the DoS problem includes a number of intruders, thus specifying DDoS attacks as well.

Definition VI.1 (Verification Scenario)**.** *A (respectively, balanced) verification scenario $\mathcal{V}$ consists of the following components:*

- *A finite set of (respectively, balanced) services $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ (see Definition IV.3);*
- *A finite set of (respectively, balanced) resource-bounded intruders $\{\mathcal{I}_1, \ldots, \mathcal{I}_m\}$;*
- *natural numbers $mdur_i$, specifying the minimal duration that the resources of the service $\mathcal{A}_i$ have to be consumed to represent a successful DoS attack.*

The initial configuration of $\mathcal{V}$, $\mathcal{S}_\mathcal{V}$, contains exactly the following timed facts, for $1 \leq i \leq n$ and $1 \leq j \leq m$:

- *$Time@0$, specifying that the initial global time is 0;*
- *$\mathsf{R}(s_i, r^s_{ini})@0$, where s_i and r^s_{ini} are the unique identification symbol and initial service resource of service $\mathcal{A}_i$;*
- *$\mathsf{Av}(s_i)@0$, specifying that the service s_i is available;*
- *$\mathsf{R}(I_j, r^{I_j}_{max})@0$, where I_j and $r^{I_j}_{max}$ are the unique identification symbol and maximal resource of the intruder $\mathcal{I}_j$;*
- *the facts in $\mathcal{M}_j$, the initial knowledge base of $\mathcal{I}_j$;*
- *for balanced verification scenario only, dum_j dummy facts $P(I_j)@0$, where dum_j is the number of available dummy facts specified in $\mathcal{I}_j$, and d_{s_i} dummy facts $D(s_i)@0$, where d_{s_i} is the number of dummy facts specified in $\mathcal{A}_i$.*

The goal of a verification scenario is as follows:

$$\mathcal{GS}_\mathcal{V} = \{\langle \{Time@T, \mathsf{Den}(s_i)@T_1\}, \{T \geq T_1 + mdur_i\} \rangle\}$$

where $1 \leq i \leq n$, and s_i is the unique identification symbol of some $\mathcal{A}_i$.

Finally, we construct the critical configuration specification of the scenario $\mathcal{CS}_\mathcal{V}$ as the union of all the critical configuration specifications of all services $\mathcal{A}_i$.

The DoS problem associated to a verification scenario is reduced to searching for non-critical traces as defined below.

Definition VI.2 (DoS Problem)**.** *Let $\mathcal{V}$ be a verification scenario. The DoS problem is to determine whether there is a non-critical trace w.r.t. $\mathcal{CS}_\mathcal{V}$ from $\mathcal{S}_\mathcal{V}$ to a goal configuration w.r.t. $\mathcal{GS}_\mathcal{V}$.*

Notice the role of protocol critical configuration specifications (Definition IV.2). Without the Timeout CS, false DoS attacks could be found as traces where the service simply does not garbage-collect protocol sessions that have expired due to a timeout. Similarly, without Denied CS and Available CS, there would be traces where the facts Av and Den are not updated according to the level of resources of the service. For example, a trace would exist with a configuration where $\mathsf{Den}(s)$ is present although the service s has enough resources.

Notice that if $mdur = 0$ in the verification scenario, we model the DoS attacks with no duration attached, as in [31].

A. Attack Example - HTTP GET

We illustrate a DoS attack on a verification scenario involving the protocol theory of the HTTP GET described in Section IV-A. The attack is very close to the Slowloris attack [43]. The values for service timeout and number of packets sent by the intruder are the same as used in practice.

Assume a single service with the HTTP GET protocol theory given in Fig. 1 with initially 300 workers available, that is, $r^s_{ini} = 300$ and $r^s_m = 0$, so the service is denied when the service has no workers left.

Consider a bounded traffic intruder I (see Section V), with the function $\textsc{spec}_{SND} = \langle 1, 30, 1\rangle$, that is, intruder consumes one resource when he sends a message and this resource can only be used again after 30 time units and $\textsc{spec}_R = \langle 0, 0, 0\rangle$ for the remaining rules R. Moreover, assume that $r^I_{max} = 350$, that is, he has 350 resources. This means that the intruder can only send 350 messages in every 30 units time window. Finally, the intruder knows the relevant information from the GET protocol, namely, the set of timed facts:

$$\begin{aligned}\mathcal{M} = \{ \quad & M(I, INIT)@0.0, M(I, GET)@0.0, \\ & M(I, Inc)@0.0, M(I, Com)@0.0\}.\end{aligned}$$

Thus, the initial configuration is:

$$\mathcal{M} \ \cup \ \{ \ Time@0, \mathsf{R}(s, 300)@0, \mathsf{R}(I, 350)@0, \mathsf{Av}(s)@0 \ \}.$$

Finally, assume that $mdur = 300$, that is, the service has to be down for 300 time units for a successful DoS attack.

The attack follows by the intruder applying the **USE** rule with $M(I, INIT)$ 300 times, that is, making 300 copies of this message. This has no cost. Then applying the **SND** rule on $M(I, INIT)$ 300 times, generating 300 copies of $\mathsf{N}(INIT)@1$. That is, the intruder sends a *burst of 300 messages*. Time then advances one time unit. Now the service applies the **INIT** rule 300 times generating 300 protocol sessions facts S_0 and consuming all the service's resources. Thus the fact $\mathsf{Av}(s)$ is replaced by $\mathsf{Den}(s)$ using the corresponding service availability rule. At this point, 30 time units pass. The intruder can then recover his resources by applying 300 instances of the **RES** rule. The intruder then applies 300 times the **USE** rule with $M(Inc)$ and applying 300 times the **SND** rule generating 300 copies of $\mathsf{N}(Inc)$. That is, the intruder sends another burst of messages. These facts are then used by the service moving to state S_1. By waiting another 30 time units, and re-generating copies of $\mathsf{N}(Inc)$, *i.e.*, sending periodic bursts of messages, the intruder is able to consume the service's resources to 0 for indefinite time, leading to a DoS attack.

Notice that this attack, while captured by our model, has a great deal of non-determinism, *e.g.*, different rules can be applied to a configuration. Moreover, the length of the witness trace is quite large, making it challenging for automated verification to find. In Section VIII, we show how Rewriting Modulo SMT [39] can help automate the search for DoS attacks by using symbolic search.

Finally, we also point out that traditional flooding attacks, such as SYN flooding can also be modelled. For such attacks, the attacker(s) has resources that allow him to send a very large number of messages.

VII. Complexity Results

Reachability and the related problems for MSR are undecidable in general [24]. However, by imposing some reasonable restrictions, such as using only balanced rules and bounding the size of facts, these problems become decidable, even in timed models with fresh values. A summary of related complexity results is shown in Table I.

The undecidability of the general version of the DoS problem follows from undecidability of secrecy problem for general MSR theories [24], [25]. We show how to encode the secrecy problem as an instance of a DoS problem in the proof of Lemma VII.2 (see Appendix A).

For the complexity of the balanced version of the DoS problem we rely on the PSPACE-completeness of the secrecy problem for bounded memory intruder and balanced MSR protocol theories [25] and on the complexity of the non-critical reachability problem for dense time MSR, which we prove to be PSPACE-complete. The technical challenges of this novel complexity result are shortly discussed below. The complete proof can be found in the technical report [23].

The following lemmas give the upper and lower bounds for the complexity of DoS problems. Proofs of these lemmas can be found in Appendix A.

Lemma VII.1. *The DoS problem is an instance of the non-critical reachability problem for MSR with real time.*

Lemma VII.2. *The secrecy problem for Dolev-Yao intruder and MSR protocol theories is an instance of the DoS problem. The secrecy problem for bounded memory Dolev-Yao intruder and balanced MSR protocol theories is an instance of the DoS problem for balanced verification scenarios.*

Theorem VII.3 (DoS problem)**.** *The DoS problem is undecidable in general.*

Theorem VII.4 (Balanced DoS problem)**.** *Assuming a bound on the size of facts, the DoS problem for balanced verification scenarios is PSPACE-complete.*

A. Complexity of non-critical reachability problem

For a given set of timed MSR rules $\mathcal{R}$, an initial configuration $\mathcal{S}_0$, a goal $\mathcal{GS}$ and a critical configuration specification $\mathcal{CS}$, *Non-critical Reachability Problem* consists in checking whether there is a non-critical trace that leads from $\mathcal{S}_0$ to a goal configuration.

TABLE I
SUMMARY OF THE COMPLEXITY RESULTS FOR THE REACHABILITY AND NON-CRITICAL REACHABILITY PROBLEMS, THAT IS, THE REACHABILITY PROBLEM INVOLVING ONLY NON-CRITICAL TRACES.

MSR		**Reachability Problem**	**Non-critical Reachability**
Balanced	untimed	PSPACE-complete [25], [28]	PSPACE-complete [25], [28]
	discrete time	PSPACE-complete [27]	PSPACE-complete [27]
	dense time	PSPACE-complete [26]	PSPACE-complete [new]
Not necessarily balanced		Undecidable [24]	Undecidable [24]

Theorem VII.5 (Non-critical Reachability Problem)**.**
The non-critical reachability problem is undecidable in general. The non-critical reachability problem for balanced timed MSR with dense time is PSPACE-complete when assuming a bound on the size of facts.

When dealing with the complexity of the verification problems in timed MSR with dense time there are several challenges that need to be addressed, starting with the underlying nondeterministic nature of the multiset rewriting formalism. Then, an unbounded number of fresh values can appear in a trace. Additionally, in our timed MSR there is no bound on the global time value, *i.e.*, on represented time periods. Furthermore, in the dense time model, there is the additional non-determinism in the choice of ε in time advancement $Tick$ rule. Finally, as discussed in Section III, the notion of non-critical traces in dense time setting is much more elaborate than in the untimed and discrete time models. Recall that, as per Definition III.1 showing that a trace of a dense time MSR is non-critical involves not only the configurations it contains, but also an infinite number of configurations obtained by decomposing $Tick$ rules in order to faithfully capture the continuity of time.

The proof of Theorem VII.5 relies on the abstractions introduced in [26], called *circle-configurations*, where it has been shown that, with respect to the reachability problem (without critical configurations) traces over circle-configurations are a sound and complete representation of traces with dense time. In particular, the $Tick$ rule is represented by a collection of rules defined over circle-configurations.

The addition of critical configurations and non-critical traces involves a new auxiliary notion of *immediate successor configurations*, related to satisfiability of relevant time constraints. Using immediate successor configurations reduces the search space when checking that a trace of dense time MSR is non-critical. Furthermore, there is only a finite number of abstractions related to a given non-critical reachability problem, which bounds the length of solution traces, resulting in a polynomial space search space. Details appear in the Technical Report [23].

VIII. AUTOMATED VERIFICATION

We describe our steps towards automation of verification, focusing on some key ideas to improve automation performance. To test these ideas, we specified verification scenarios for the Slowloris and Slow-TCAM attacks. The implementation is available at [1].[4]

As illustrated by example in previous sections, MSR rules generally lead to an enormous search space which makes traditional automated verification impractical. For example, to find the Slowloris attack in the scenario used in the example in Section VI-A, one would need to search a tree of depth of at least 500. Moreover, it is not possible to instantiate all values of ε in the time advancement rule as there are infinite possibilities.

To reduce search space, we use symbolic search through Rewriting Modulo SMT [39]. This allows us to perform symbolic search relying on the power of off-the-shelf SMT solvers. We considered three types of symbols:

Time Symbols: Instead of using concrete values for global time, we use time symbols. Time symbols, ts, may be used in expressions, such as in $\mathrm{ts} + 2.0$;

Intruder and Service Resource Symbols: Instead of using concrete values for intruder and service resources, we use intruder resource symbols and service resource symbols;

Protocol Instance Symbols: Instead of creating one protocol session, we allow the intruder to create several instances of a protocol session representing a burst from the intruder. However, we use symbolic values for the number of instances created.

The symbolic rewrite rules accumulate constraints on the values and search stops whenever the set of accumulated constraints is not satisfiable.

The intruder model used is based on the intruder model proposed here, but weaker, limited to sending a pre-defined set of messages. This limitation may be overcome by also representing messages symbolically.

Table II summarizes our preliminary results using our symbolic machinery. We additionally considered bounded model-checking by bounding the following parameters. Thus, our method, while sound, it is not complete.

Number of Parallel Symbolic Protocols (pxs)**:** Allowing the intruder to create an unbounded number of sessions increases greatly the size of search space. We thus bound the number of parallel symbolic protocols. Notice that this does not mean that we are bounding the number of parallel sessions because symbolic protocols still carry the number of instance symbol;

[4] Notice that the intruder model introduced here is different from the usual Dolev-Yao like intruder models used in existing protocol security verification tools, as the latter does not explicitly mention resources nor timeouts. Therefore, it is not clear how they can be used to automatically verify systems.

TABLE II
Automated verification results using a 2.7 GHz machine with 8 GB memory. SL [i] is Slowloris with $mdur$ of i timeouts. STCAM [i] is a scenario where the intruder requires i symbolic protocol sessions in parallel to carry out a Slow-TCAM attack. Search was interrupted after 10 mins.

	No Bounding		**Bounded** msgs_1		**Bounded** pxs		**Bounded** msgs_1 **and** pxs	
Attack	States	Time (s)	States	Time (s)	States	Time (s)	States	Time (s)
SL [1]	18	0.4	16	0.4	8	0.1	7	0.1
SL [2]	409	13	277	11.2	27	0.4	17	0.4
SL [3]	–	–	–	–	228	4.7	56	2.6
STCAM [2]	17	0.3	15	0.3	16	0.3	14	0.2
STCAM [3]	387	12.5	266	9.7	361	10.3	243	9.2
STCAM [4]	–	–	–	–	12783	561	6322	474

Number of Different Types of Messages at a Time (msgs_1)**:** Messages of different types may be created. The greater the types of messages the greater is the interleaving causing state space to increase. We, therefore, bounded the number of types of messages that can be generated.

We are able to discover the Slowloris attack using different values for $mdur$ up to 3 times the protocol timeout. This information can be used by specifiers to build defenses to mitigate such attacks. Moreover, we were able to discover the SlowTCAM attack. We varied the power of the service while using the same intruder. We have found the non-trivial attacks where the intruder maintains protocol sessions due to 2 or 3 bursts, *i.e.*, 2 or 3 symbolic protocol sessions. This information can be used by specifiers to generalize attacks to greater instances and investigate adequate defenses.

While these bounds (on parallel symbolic protocols and messages to be processed) do not affect the soundness of our approach as we are simply bounding the search engines choices, it may affect the range of attacks that can be found. For example, attacks that require more symbolic parallel protocols than the bound would not be found. Indeed, setting this bound to one, we were not able to find the SlowTCAM nor the PRA attacks. Similarly, in order to discover a wider range of attacks or verify security of a protocol, sufficient details about the server and the protocol execution should be included in the model.

IX. Conclusions and Related Work

This paper introduced a new framework for analyzing the security of systems against DoS attacks. The framework allows reasoning about service's and intruder's resources and service timeouts. We illustrate the power of the model with a number of examples of attacks and intruder models. We study the complexity of the DoS problem, showing it to be undecidable in general and PSPACE-complete for balanced verification scenarios. For the latter results, we provide new results on the complexity of the non-critical reachability problem. Finally, we demonstrate the use of Rewriting Modulo SMT for efficiently automating the verification task.

While our work is inspired by the work [31], in contrast to the model [31], our model mentions time explicitly. This means that our model can reason about service timeouts, which are essential for discovering vulnerabilities to Slow and Asymmetric Attacks. Moreover, this leads to a more refined definition of DoS attacks than the one proposed in [31]. Intuitively, a DoS attack is defined as exhausting the target service's resource for a certain period of time. This duration can be specified in our model with explicit time, thus reflecting the intuitive notion of a DoS attack, which is to take down a service temporarily or indefinitely. Finally, our model can also specify time-based counter-measures, that issue timeouts whenever some condition is applicable.

In [32] a taxonomy of DoS attacks and defense mechanisms is presented. Some of the relevant security issues have clearly been covered in our model. For example, attack rate dynamics issue resulting in constant or variable rate attacks is captured by recovery of resources within associated time through R and Rec facts. We, however, did not go into details of all issues identified in [32], *e.g.*, means used to prepare and perform the attack (manual, semi-automatic and automatic DDoS attacks) nor into source address validity issue (spoofed vs. valid IPs), impact on the victim (recoverable vs. non-recoverable attacks) etc. Nevertheless, additional details could be introduced in the model in relation to such security issues. By including various resources of the service in the specification, we are able to represent and differentiate between flooding attacks and more sophisticated semantic attacks. This also applies to modelling of DoS defense mechanisms.

Authors in [42] demonstrate a model checking technique, called measure checking, for finding amplification attacks on VoIP using rewriting logic (implemented in Maude). They do not provide, however, general intruder models based on the DY intruder as we do, nor the corresponding complexity results. Finally, we also consider a wider range of attacks, such as slow and CPU exhaustion attacks.

MSR frameworks [4], [16], [25], [26] have been proposed for security verification. However, they were interested in authentication and secrecy-related problems, and distance-bounding protocols and not in DoS attacks, which is our main goal here. Moreover, we also extend the model and the complexity results of [26] by considering the non-critical reachability problem and not only the simple reachability problem. We also build on the work of [25] which considers bounded memory intruders. Our intruders can be bounded with respect to a wider range of types of resources.

Timed automata (TA) [5] have been used for the verification of many systems involving real-time. The framework we propose is more closely related to security, as resources, intruder models, and DoS problems are considered. This is obtained by means of adding explicit notions of timeouts and also of resources to rules. Using first-order rules in MSR versus finite number of states and rules in TA, further affects comparisons of complexity results. Also, our model has the additional feature of non-critical traces, distinguishing among potential reachability solutions only those traces that have no negative properties, *i.e.*, are non-critical.

Protocol verification in [22], using timed automata, also involves timing aspects of security protocols in the presence of DY intruder, as well as automated tools. They investigate timed authentication properties, based on expected time intervals for completion of successful protocol sessions. Such an approach may not be as adequate for our DoS problems, as the service resources may also be consumed by sessions that have not successfully completed. Also, we consider more general protocol theories with varied execution time of a correct sessions, due to, *e.g.*, possibility of sending incomplete headers in a correct protocol execution, or even loops in protocols which are, differently from [22], allowed in our protocol theories etc. Also, we investigate the computational complexity related to the verification.

A decidability result relating to timing attacks in security protocols is given in [11]. The result is based on symbolic equivalence of traces, applied, in particular on verification of privacy properties, not DoS. Similar to our parametrized action execution, formalized using SPEC function, they also deal with computation time w.r.t. to length of inputs. The model allows representation of other "side-channel" resources that can be leaked by the execution, such as power consumption. As this approach is based on the reduction of time trace equivalence to length trace equivalence, it remains to be investigated whether such an approach may be applicable to the general DoS problem, covering protocol theories with varied execution time, as already discussed above.

Statistical Model Checking has been used to investigate the effectiveness of attack defense [2], [3], [13], [17], [18], [30]. We believe that the search space reduction due to the use of symbolic search can improve the performance of these methods for the verification of defense. This is left for future work.

Also, we expect that in our model we are able to capture a larger class of security problem closely related to DoS attacks, including other types of attacks that may include a DoS as a component, attempts to prevent a particular individual from accessing a service, attempts to disrupt service to a specific system or person, as well as poor service performance resulting from intruder interference.

Finally, we would like to investigate how different resource-bounded intruder models can be compared. For example, whether it is possible to define a partial order relating the strength of intruders. This is left for future work.

Acknowledgments

Part of this work was done during the visits to the University of Pennsylvania by Alturki, Ban Kirigin, Kanovich, Nigam, and Talcott, which were partially supported by ONR grant N00014-15-1-2047 and by the University of Pennsylvania. Ban Kirigin is supported in part by the Croatian Science Foundation under the project UIP-05-2017-9219. Scedrov is partially supported by ONR grants N00014-15-1-2047 and N00014-18-1-2618. The participation of Kanovich and Scedrov in the preparation of this article was partially within the framework of the HSE University Basic Research Program funded by the Russian Academic Excellence Project '5-100'. Talcott is partly supported by ONR grant N00014-15-1-2202 and NRL grant N0017317-1-G002. Nigam is partially supported by NRL grant N0017317-1-G002, and CNPq grant 303909/2018-8.

References

[1] Resource bounded intruder implementation in Maude https://github.com/viveknigam/boundedIntruder. 2018.

[2] M. AlTurki, J. Meseguer, and C. A. Gunter. Probabilistic modeling and analysis of DoS protection for the ASV protocol. *Electr. Notes Theor. Comput. Sci.*, 234:3–18, 2009.

[3] M. A. AlTurki, M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, and C. Talcott. Statistical model checking of distance fraud attacks on the Hancke-Kuhn family of protocols. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 60–71. ACM, 2018.

[4] M. A. AlTurki, M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, and C. Talcott. A multiset rewriting model for specifying and verifying timing aspects of security protocols. http://nigam.info/docs/CM_timed_intruders.pdf, 2019.

[5] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183 – 235, 1994.

[6] E. Cambiaso, G. Papaleo, and M. Aiello. Slowdroid: Turning a smartphone into a mobile attack vector. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 405–410. IEEE, 2014.

[7] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello. Slow DoS attacks: definition and categorisation. *International Journal of Trust Management in Computing and Communications*, 1(3-4):300–319, 2013.

[8] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello. Mobile executions of slow DoS attacks. *Logic Journal of IGPL*, pages 54–67, 2015.

[9] CERT Coordination Center, Denial of Service Attacks. www.cs.columbia.edu/~danr/courses/6761/Fall00/week14/cert.ps.

[10] I. Cervesato, N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *CSFW*, pages 55–69, 1999.

[11] V. Cheval and V. Cortier. Timing attacks in security protocols: symbolic framework and proof techniques. In *International Conference on Principles of Security and Trust*, pages 280–299. Springer, 2015.

[12] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer, 2007.

[13] Y. G. Dantas, V. Nigam, and I. E. Fonseca. A selective defense for application layer DDoS attacks. In *IEEE JISIC 2014*, pages 75–82, 2014.

[14] Department of Homeland Security, Understanding Denial-of-Service Attacks. https://www.us-cert.gov/ncas/tips/ST04-015.

[15] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[16] N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[17] J. Eckhardt, T. Mühlbauer, M. AlTurki, J. Meseguer, and M. Wirsing. Stable availability under denial of service attacks through formal patterns. In *FASE*, pages 78–93, 2012.

[18] J. Eckhardt, T. Mühlbauer, J. Meseguer, and M. Wirsing. Statistical model checking for composite actor systems. In *WADT*, pages 143–160, 2012.

[19] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

[20] P. Gupta and V. Shmatikov. Security analysis of voice-over-ip protocols. In *20th IEEE Computer Security Foundations Symposium, Venice, Italy*, pages 49–63. IEEE Computer Society, 2007.

[21] IETF. [TLS] SSL Renegotiation DOS. Available at https://www.ietf.org/mail-archive/web/tls/current/msg07553.html.

[22] G. Jakubowska and W. Penczek. Modelling and checking timed authentication of security protocols. *Fundamenta Informaticae*, 79(3-4):363–378, 2007.

[23] M. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, and C. Talcott. Compliance in real time multiset rewriting models. Available at https://arxiv.org/abs/1811.04826.

[24] M. Kanovich, P. Rowe, and A. Scedrov. Policy compliance in collaborative systems. In *CSF '09: Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*, pages 218–233, Washington, DC, USA, 2009. IEEE Computer Society.

[25] M. I. Kanovich, T. B. Kirigin, V. Nigam, and A. Scedrov. Bounded memory Dolev-Yao adversaries in collaborative systems. *Inf. Comput.*, 238:233–261, 2014.

[26] M. I. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, and C. L. Talcott. Time, computational complexity, and probability in the analysis of distance-bounding protocols. *Journal of Computer Security*, 25(6):585–630, 2017.

[27] M. I. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, C. L. Talcott, and R. Perovic. A rewriting framework and logic for activities subject to regulations. *Mathematical Structures in Computer Science*, 27(3):332–375, 2017.

[28] M. I. Kanovich, P. Rowe, and A. Scedrov. Collaborative planning with confidentiality. *J. Autom. Reasoning*, 46(3-4):389–421, 2011.

[29] M. O. O. Lemos, Y. G. Dantas, I. Fonseca, V. Nigam, and G. Sampaio. A selective defense for mitigating coordinated call attacks. In *34th Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, 2016.

[30] M. O. O. Lemos, Y. G. Dantas, I. E. Fonseca, and V. Nigam. On the accuracy of formal verification of selective defenses for TDoS attacks. *J. Log. Algebr. Meth. Program.*, 94:45–67, 2018.

[31] C. A. Meadows. A cost-based framework for analysis of denial of service networks. *Journal of Computer Security*, 9(1/2):143–164, 2001.

[32] J. Mirkovic and P. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[33] V. Nigam, C. Talcott, and A. A. Urquiza. Towards the automated verification of cyber-physical security protocols: Bounding the number of timed intruders. In *European Symposium on Research in Computer Security (ESORICS)*, 2016.

[34] O. Olivo, I. Dillig, and C. Lin. Detecting and exploiting second order denial-of-service vulnerabilities in web applications. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 616–628, New York, NY, USA, 2015. ACM.

[35] OpenFlow. *Open Networking Foundation (ONF)*. https://www.opennetworking.org/. Accessed in: 02 de Setembro de 2016.

[36] T. A. Pascoal, Y. G. Dantas, I. E. Fonseca, and V. Nigam. Slow TCAM exhaustion DDoS attack. In *ICT Systems Security and Privacy Protection (IFIP SEC)*, 2017.

[37] r-u-dead yet. https://code.google.com/p/r-u-dead-yet/. 2013.

[38] ReQTimeOut. https://httpd.apache.org/docs/2.4/mod/mod_reqtimeout.html. 2014.

[39] C. Rocha, J. Meseguer, and C. A. Muñoz. Rewriting modulo SMT and open system analysis. *J. Log. Algebr. Meth. Program.*, 86(1):269–297, 2017.

[40] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393.

[41] P. Rowe. *Policy compliance, confidentiality and complexity in collaborative systems*. PhD thesis, University of Pennsylvania, 2009.

[42] R. Shankesi, M. AlTurki, R. Sasse, C. A. Gunter, and J. Meseguer. Model-checking DoS amplification for VoIP session initiation. In *ESORICS*, pages 390–405, 2009.

[43] slowloris. http://ha.ckers.org/slowloris/. 2013.

[44] slowread. https://code.google.com/p/slowhttptest/. 2013.

[45] R. Sparks, S. Lawrence, A. Hawrylyshen, and B. Campen. Addressing an Amplification Vulnerability in Session Initiation Protocol (SIP) Forking Proxies. RFC 5393 (Proposed Standard), Dec. 2008.

[46] B. Sullivan. Application-level denial of service attacks and defenses, 2011. [Online; Accessed 16-December-2016].

Appendix

A. DoS Problem PSPACE Complexity Proofs

Proof of Lemma VII.1

Proof. Given a verification scenario $\mathcal{V}$, with all the notation as per Definition VI.1, we construct an instance of the non-critical reachability problem, $\mathcal{D}$, such that $\mathcal{D}$ has a solution trace $\mathcal{T}$ iff $\mathcal{V}$ is vulnerable to a DoS attack as per witness trace $\mathcal{T}$. This follows easily from the definition of the problem $\mathcal{D}$, by inspecting the protocol and intruder rules, critical and goal configurations.

We set the non-critical reachability problem $\mathcal{D}$ as follows. We set MSR rules $\mathcal{R}$ to consist of intruder rules $\mathcal{I}_j, 1 \leq j \leq m$, and rules of protocol theories $\mathcal{A}_i, 1 \leq i \leq n$. The initial coniguration of $\mathcal{D}$ is set to be $\mathcal{S}_0$, the inital configuration of verification scenario $\mathcal{V}$. Also, the goal of $\mathcal{D}$ is specified by the goal of $\mathcal{V}$:

$$\langle\ Time@T, \mathsf{Den}(s_i)@T',\ T - T' \geq d_{min}\rangle\ .$$

We set critical configuration specification $\mathcal{CS}$ of $\mathcal{D}$ as $\mathcal{CS}_{\mathcal{V}}$, *i.e.*, to consist of protocol $\mathcal{CS}$es of $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Notice that, as per protocol critical configuration specifications, $\mathcal{CS}$ of $\mathcal{D}$ contains:
$\langle Time@T, \mathsf{S}_i(S_{id}, R, \vec{x}_i)@T' \mid T' < T\rangle$, for all $i > 0$, forcing timeouts of protocol states, and
$\langle\ \mathsf{R}(S_{id}, r_m^{S_{id}})@T, \mathsf{Av}(S_{id})@T' \mid T > T'\rangle$ and
$\langle\ \mathsf{R}(S_{id}, R + 1 + r_m^{S_{id}})@T, \mathsf{Den}(S_{id})@T' \mid T > T'\rangle$
for the auxiliary flags, Den, Av, used for modelling insufficient server resources. □

Proof of Lemma VII.2

Proof. Secrecy problem of whether or not an intruder can discover a secret originally known only to another protocol participant, is undecidable in general [16], and PSPACE-complete for bounded memory intruder and balanced MSR protocol theories [25].

We encode the secrecy problem as an instance of a DoS problem. In particular, we specify that a DoS problem can occur only if the intruder discovers the secret.

We describe the case of balanced intruder and protocol models. The general case follows by simply ignoring dummy facts.

Let $\mathcal{S}_0$ be the initial configuration of the given secrecy problem $\mathcal{D}$. We assume $\mathcal{S}_0$ to contain the facts representing initial knowledge, such as participants names and keys, a fact denoting that a secret α is known to some participant, as well as the initial intruder knowledge and a number of dummy facts P and D representing intruder and system memory, respectively. Let $\mathcal{R}$ be the given protocol theory. Its rules have one of the following forms:

$$\begin{array}{l}
W\ \ P(*) \rightarrow W\ \ S_0(\vec{x}) \\
S_0(\ldots)\ P(*)\ W \rightarrow \exists \vec{z}. S_l(\ldots)\ N(\ldots)\ W' \\
S_i(\ldots)\ N(\ldots)\ W \rightarrow \exists \vec{z}. S_j(\ldots)\ N(\ldots)\ W' \\
S_h(\ldots)\ N(\ldots)\ W \rightarrow \exists \vec{z}. S_k(\ldots)\ P(*)\ W' \\
S_k \rightarrow P(*)
\end{array} \tag{3}$$

where $l > 0$, $j > i$, $k > h$, S_k is the final state of one of the protocol role theories, and W and W' are multisets of facts containing no role states nor any N facts.

Intruder theory $\mathcal{I}$ is given as bounded memory intruder rules depicted in Fig. 4.

I/O Rules:

REC: $N(x)\ R(*) \rightarrow D(x)\ P(*)$

SND: $C(x)\ P(*) \rightarrow N(x)\ R(*)$

Composition and Decomposition Rules:

COMP: $M(x)\ M(y) \rightarrow M(\langle x, y\rangle)\ P(*)$

DCMP: $M(\langle x, y\rangle)\ P(*) \rightarrow M(x)\ M(y)$

USE: $M(x)\ P(*) \rightarrow M(x)\ M(x)$

ENC: $KP(k_d, k_e)\ M(k_e)\ M(x) \rightarrow$
$KP(k_d, k_e)\ M(k_e)\ M(enc(k_e, x))$

DEC: $M(k_d)\ KP(k_e, k_d)\ M(enc(k_e, x))\ R(*)$
$\rightarrow M(k_d)\ KP(k_e, k_d)\ M(x)\ M(enc(k_e, x))$

GEN: $P(*) \rightarrow \exists n. M(n)$

Memory maintenance rules:

DELM: $M(x) \rightarrow P(*)$

Fig. 4. Bounded Memory Intruder Theory

I/O Rules:

REC: $\mathit{Time}@T, N(X)@T_1, \mathsf{R}(I_{id}, Z + r_R)@T_2, P(I_{id})@T_3 \mid T \geq T_1 \longrightarrow$
$\mathit{Time}@T, M(I_{id}, X)@(T + \delta_L), \mathsf{R}(I_{id}, Z)@T, \mathsf{Rec}(i_{id}, r_R)@(T + \delta_R)$
where $\textsc{spec}_{REC}(X, I_{id}) = \langle \delta_L, \delta_R, r_R \rangle$

SND: $\mathit{Time}@T, M(I_{id}, X)@T_1, \mathsf{R}(I_{id}, Z + r_R)@T_2, P(I_{id})@T_3 \mid T \geq T_1 \longrightarrow$
$\mathit{Time}@T, N(X)@(T + \delta_L), \mathsf{R}(I_{id}, Z)@T, \mathsf{Rec}(I_{id}, r_R)@(T + \delta_R)$
where $\textsc{spec}_{SND}(X, I_{id}) = \langle \delta_L, \delta_R, r_R \rangle$

Message Composition and Decomposition Rules:

COMP: $\mathit{Time}@T, M(I_{id}, X)@T_1, M(I_{id}, Y)@T_2, \mathsf{R}(I_{id}, Z + r_R)@T_3 \longrightarrow$
$\mathit{Time}@T, M(I_{id}, \langle X, Y \rangle)@(T + \delta_L), \mathsf{R}(I_{id}, Z)@T, \mathsf{Rec}(I_{id}, r_R)@(T + \delta_R)$
where $\textsc{spec}_{COMP}(X, Y, I_{id}) = \langle \delta_L, \delta_R, r_R \rangle$

DCMP: $\mathit{Time}@T, M(I_{id}, \langle X, Y \rangle)@T_1, \mathsf{R}(I_{id}, Z + r_R)@T_2, P(I_{id})@T_3, P(I_{id})@T_4 \longrightarrow$
$\mathit{Time}@T, M(I_{id}, X)@(T + \delta_L), M(I_{id}, Y)@(T + \delta_L)\mathsf{R}(i_{id}, Z)@T, \mathsf{Rec}(i_{id}, r_R)@(T + \delta_R)$
where $\textsc{spec}_{DCMP}(\langle X, Y \rangle, i_{id}) = \langle \delta_L, \delta_R, r_R \rangle$

USE: $\mathit{Time}@T, M(I_{id}, X)@T_1, \mathsf{R}(I_{id}, Z + r_R)@T_2, P(I_{id})@T_3, P(I_{id})@T_4 \longrightarrow$
$\mathit{Time}@T, M(I_{id}, X)@T_1, M(I_{id}, X)@(T + \delta_L), \mathsf{R}(I_{id}, Z)@T, \mathsf{Rec}(I_{id}, r_R)@(T + \delta_R),$
where $\textsc{spec}_{USE}(X, I_{id}) = \langle \delta_L, \delta_R, r_R \rangle$

ENC: $\mathit{Time}@T, M(I_{id}, K)@T_1, M(I_{id}, X)@T_2, \mathsf{R}(I_{id}, Z + r_R)@T_3, P(I_{id})@T_4, P(I_{id})@T_5 \longrightarrow$
$\mathit{Time}@T, M(I_{id}, K)@T_1, M(I_{id}, X)@T_2, M(I_{id}, \{X\}_K)@(T + \delta_L),$
$\mathsf{R}(I_{id}, Z)@T, \mathsf{Rec}@(I_{id}, r_R)(T + \delta_L)$
where $\textsc{spec}_{ENC}(K, X, I_{id}) = \langle \delta_L, \delta_R, r_R \rangle$

DEC: $\mathit{Time}@T, M(I_{id}, K^{-1})@T_1, M(I_{id}, \{X\}_K)@T_2, \mathsf{R}(I_{id}, Z + r_R)@T_3, P(I_{id})@T_4, P(I_{id})@T_5 \longrightarrow$
$\mathit{Time}@T, M(I_{id}, K^{-1})@T_1, M(I_{id}, \{X\}_K)@T_2, M(I_{id}, X)@(T + \delta_L),$
$\mathsf{R}(I_{id}, Z)@T, \mathsf{Rec}(I_{id}, r_R)@(T + \delta_R)$
where $\textsc{spec}_{DEC}(K^{-1}, \{X\}_K, I_{id}) = \langle \delta_L, \delta_R, r_R \rangle$

GEN: $\mathit{Time}@T, \mathsf{R}(I_{id}, Z + r_R)@T_1, P(I_{id})@T_2, P(I_{id})@T_3 \longrightarrow$
$\exists N.\mathit{Time}@T, M(I_{id}, N)@(T + \delta_L), \mathsf{R}(I_{id}, Z)@T, \mathsf{Rec}(I_{id}, r_R)@(T + \delta_R)$
where $\textsc{spec}_{GEN}(I_{id}) = \langle \delta_L, \delta_R, r_R \rangle$

Maintenance Rules:

RES: $\mathit{Time}@T, \mathsf{R}(I_{id}, Z)@T_1, \mathsf{Rec}(I_{id}, r_R)@T_2 \mid T_2 \leq T \longrightarrow \mathit{Time}@T, \mathsf{R}(I_{id}, Z + r_R)@T, P(I_{id})@T$

DELM: $\mathit{Time}@T, M(I_{id}, X)@T_1 \longrightarrow \mathit{Time}@T, P(I_{id})@T$

Fig. 3. Balanced Bounded Resource Intruder Theory

The intruder $\mathcal{I}$ naturally corresponds the balanced resource-bounded intruder $\mathcal{I}'$ with an identifier i, as shown in Fig. 3, with $\textsc{spec}_R = \langle 0, 0, 0 \rangle$, for all intruder rules R. Notice that the intruder spends no resources for his actions.

To the above given secrecy problem we relate the following verification scenario $\mathcal{V}$.

Protocol resource theory of the service with identifier s is obtained by translating the above rules (3), simply by adding *e.g.*, resource facts R, obtaining thus protocol initialization and execution rules, and by adding the protocol state timeout and service availability rules, as per Definition IV.1. All the facts in the obtained rules are timestamped with time variables, the fact $\mathit{Time}@T$ is added, as well as the corresponding time constraints, as per Definition IV.1. Here we set all state timeout values to 1, $r^s_m = 0$ as well as having initial resources $r^s_{ini} = 1$, and 0 initialization and execution costs for all the rules. We also set $\mathit{mdur} = 0$.

Initial configuration contains all the facts of $\mathcal{S}_0$ timestamped with 0, and additional facts $\mathit{Time}@0$, $\mathit{Av}(s)@0$, $R(s, 1)@0$ and $R(i, 0)@0$.

We finally add a special protocol resource theory of service s that is enabled whenever the secret α is released on the network:

$$\begin{array}{c}\mathit{Time}@T, \mathsf{R}(s, R + 1)@T_1, \mathsf{N}(\alpha)@T_2 \ \mid \ T_2 \leq T \longrightarrow \\ \exists S_{id}.[\mathit{Time}@T, \mathsf{S}_0(s, S_{id}, 1)@(T + 1), \mathsf{R}(s, R)@T]\end{array} \quad (4)$$

Notice that this rule is the only rule with non-zero cost. It has the cost 1, and is applicable only when intruder knows the secret α.

Therefore, it follows that the secrecy problem $\mathcal{D}$ has a solution (*i.e.*, the secret is released onto the network) iff above rule (4) is followed by the service availability rule (obtaining a fact $\mathsf{Den}(s)$), that is, because mdur is 0, iff the related scenario $\mathcal{V}$ allows a DoS attack. □